Cisco ACNS Software Configuration Guide
for Locally Managed Deployments

Release 5.5

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
Preface 21
   Document Objectives 21
   Audience 22
   Document Organization 23
   Document Conventions 26
   Related Documentation 27
   Obtaining Documentation 27
      Cisco.com 27
      Product Documentation DVD 27
      Ordering Documentation 28
   Documentation Feedback 28
   Cisco Product Security Overview 28
      Reporting Security Problems in Cisco Products 29
   Obtaining Technical Assistance 29
      Cisco Technical Support & Documentation Website 29
      Submitting a Service Request 30
      Definitions of Service Request Severity 30
   Obtaining Additional Publications and Information 31

PART 1 Overview

CHAPTER 1 Introduction 1
   Overview of the ACNS Software Solution 1
      Types of ACNS Network Devices 3
         About Standalone Content Engines 4
      Types of Content Served in ACNS Networks 5
   Overview of Standalone Content Engine Capabilities 6
      Proxying Services with Standalone Content Engines 7
         Forward Proxy Caching 7
         Reverse Proxy Caching 8
      Caching and Streaming Services with Standalone Content Engines 9
         Supported Protocols for Caching and Streaming 12
         Web Clients Supported by Standalone Content Engines 12
         Supported Caching and Streaming Services with Direct Proxy Routing 13
**Contents**

- Supported Caching and Streaming Services with Transparent Redirection 14
- Filtering and Access Control with Standalone Content Engines 15
- Authentication, Authorization, and Accounting with Standalone Content Engines 16
- Monitoring and Troubleshooting Features with Standalone Content Engines 17

---

### CHAPTER 2

**Understanding the Basics** 1

- Key Terminology 2
- Overview of WCCP 6
  - Overview of WCCP Services 8
- Understanding Some Basic ACNS Software Caching Concepts 8
  - Introduction to Caching 8
  - Cacheable Content 10
  - Network Protocols and Caching 10
- Understanding Some Basic ACNS Streaming Media Concepts 10
  - Understanding Streaming Media Solutions 11
  - How Client Browsers Start Media Players on Client Desktops 13
  - How a Client Media Player Issues a Request 14
  - Understanding Caching Policies in Streaming Media Caching 15
- Understanding IP Multicasting Fundamentals 16
  - Insecure Services 16
  - Copying Files Between Servers and Clients 16
  - Limited Scope Addresses 16
  - Source-Specific Multicast Addresses 16
  - GLOP Addresses 17
  - Layer 2 Multicast Addresses 17

---

### CHAPTER 3

**Deployment Scenarios for Standalone Content Engines** 1

- Deciding Which Services to Deploy 2
- Deploying Caching and Streaming Services in Nontransparent Mode 3
  - Overview of Nontransparent Forward Proxy Caching 3
    - Configuring Incoming Proxy Ports for Nontransparent Mode Services 4
    - SSL Tunneling and Nontransparent Caching Deployments 6
- Deploying Caching and Streaming Services in Transparent Mode 6
  - Overview of Transparent Forward Proxy Caching 6
    - Transparent Redirection and Forward Proxy Caching with a Layer 4 Switch 8
    - Transparent Redirection and Forward Proxy Caching with a WCCP Router 9
  - Overview of Transparent Reverse Proxy Caching 10
    - Example 1—Deploying Reverse Proxy Caching with WCCP Transparent Redirection 12
    - Example 2—Deploying Reverse Proxy Caching with Layer 4 Switching 13
## Contents

- Using Advanced Transparent Caching Features 14
- Deploying Streaming Media Services on Standalone Content Engines 14
- Deploying Filtering and Access Control Services 15

### Part 2

### Basic Configuration for Standalone Content Engines

#### Chapter 4

- Getting Started 1

  - Overview of Configuring, Monitoring, and Troubleshooting Standalone Content Engines 2
  - Flowcharts of Configuring, Monitoring, and Troubleshooting Standalone Content Engines 2
  - Checklist for Configuring, Monitoring, and Troubleshooting Standalone Content Engines 7

  - Configuring a Basic Configuration on Standalone Content Engines with the Setup Utility 10
  - Commonly Used Caching Services Configurable Through the Setup Utility 11

  - Using the Setup Utility 12
    - Setup Utility Menu Options and Corresponding CLI Commands 14
    - Setup Utility Arrows and Keys 15

  - Deciding the Addressing Scheme for Standalone Content Engines 16

  - About Device Network Settings for Standalone Content Engines 17
    - Using the CLI Command Method to Configure General Settings for Standalone Content Engines 18

  - Launching the Setup Utility 19
    - Manually Launching the Setup Utility 19
    - Automatically Launching the Setup Utility 20

  - Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine 21

  - Configuring Client Browsers and Media Players for Direct Proxy Routing 35
    - Pointing Client Browsers Directly to a Standalone Content Engine 36
    - Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine 37
    - Manually Pointing Client Browsers to a Standalone Content Engine 42

    - Video Streaming: Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests 43
    - Video Streaming: Windows Media Players Directly to a Standalone Content Engine for WMT MMS Requests 45

    - Pointing RealMedia Players Directly to a Standalone Content Engine 46

  - Configuring WCCP Routers for Transparent Redirection 47

  - Verifying the Basic Configuration 47

  - Modifying the Basic Configuration Through the Setup Utility 50

  - Logging in to Standalone Content Engines 50
    - Using Telnet or a Console Session to Log in to a Standalone Content Engine 50
    - Using Secure Shell Version 1 or Version 2 to Log in to a Standalone Content Engine 52
    - Secure File Transfer Protocol Access for Nonadministrative Users 53
Performing Other Basic Tasks for Standalone Content Engines

- Showing Inventory
- Managing Administrative Login Accounts
- Setting the System Clock
  - Displaying the Standard Time Zones
- Configuring Banners for Standalone Content Engines
  - Adding or Modifying Administrative Login Accounts
- Configuring Disk Space
  - Creating Disk Space for the Sysfs
- Removing All Disk Partitions on a Single Disk Drive
- Stopping Applications from Using a Disk Drive
- Displaying the Current Disk Configuration
- Mounting to a Network Attached Storage Device
- Saving the Current Configuration on Standalone Content Engines
- Disabling Transparent Caching Services on Standalone Content Engines
- Creating Custom Message Pages for Standalone Content Engines
  - Creating HTTP Custom Error Pages for Standalone Content Engines
  - Creating a Custom Error-Signature Message for HTTP Custom Error Pages
  - Creating Custom Messages for FTP Proxy Responses for FTP Native Requests
- Shutting Down Standalone Content Engines
  - Shutting Down Standalone Content Engines from the Command Line
  - Shutting Down and Rebooting Standalone Content Engines from the Content Engine GUI
- Removing or Replacing a Content Engine
- Remotely Upgrading the BIOS

Configuring Transparent Redirection for Standalone Content Engines

- Overview of WCCP Transparent Redirection
- Redirection of Packets with WCCP
  - Redirection of Packets with Content Engine Clusters and WCCP-Enabled Routers
- About Dynamic Load Distribution with WCCP Version 2
- About Packet-Forwarding Methods
- Using Layer 3 GRE as a Packet-Forwarding Method
- Using Layer 2 Redirection as a Packet-Forwarding Method
Configuring Standalone Content Engines for WCCP Transparent Redirection 9

| Enabling WCCP on Standalone Content Engines | 12 |
| Defining Port Lists on Standalone Content Engines | 12 |
| Defining Router Lists on Standalone Content Engines | 12 |
| Configuring WCCP Services on Standalone Content Engines | 14 |

Configuring Standalone Content Engines to Support User-Defined WCCP Services 15

Displaying WCCP Configuration Information for Standalone Content Engines 16

Disabling and Reenabling WCCP Flow Redirection on Standalone Content Engines 17

Shutting Down WCCP on Standalone Content Engines 17

Configuring a Router for WCCP Transparent Redirection 18

| Setting a Password for a WCCP Version 2-Enabled Router | 20 |
| Performing a General WCCP Version 2 Configuration on a Router | 20 |
| Enabling WCCP on a Router | 20 |
| Enabling a WCCP Version 2 Router to Support WCCP Service Groups | 20 |
| Enabling WCCP Redirection on the Router | 23 |
| Enabling Packet Redirection on Outbound or Inbound Interfaces Using WCCP | 24 |
| Bypassing the Content Engine with Router Access Lists | 26 |

Configuring WCCP Services on a Router 27

| Configuring the Standard Web-Cache Service (Service 0) on a Router | 27 |
| Configuring the DNS Caching Service (Service 53) on a Router | 28 |
| Configuring the FTP-Native Caching Service (Service 60) on a Router | 29 |
| Configuring the HTTPS-Cache Service (Service 70) on a Router | 29 |
| Configuring the RTSP Service (Service 80) on a Router | 30 |
| Configuring the WMT-RTSPU Service (Service 83) on a Router | 31 |
| Configuring User-Defined WCCP Services (Services 90–97) on a Router | 31 |
| Configuring the Custom-Web-Cache Service (Service 98) on a Router | 32 |
| Configuring the Reverse-Proxy Service (Service 99) on a Router | 33 |

Clearing WCCP Statistics on a Router 34

Configuring WCCP Layer 2 Support 34

| Configuring Layer 2 Forwarding with the Hash Load-Balancing Method | 35 |
| Configuring Layer 2 Forwarding with the Mask Load-Balancing Method | 36 |

Examples of Configuring WCCP Services for Standalone Content Engines 38

| Example 1—Configuring the Web-Cache Service with WCCP Version 1 | 39 |
| Example 2—Configuring the Web-Cache Service with WCCP Version 2 | 43 |
| Example 3—Configuring the HTTPS Transparent Caching Service with WCCP Version 2 | 45 |
| Example 4—Configuring Multiple WCCP Version 2 Services on Standalone Content Engines | 46 |

Configuring Layer 4 Switching as a Redirection Method 50

| Example of Configuring Transparent Caching Using Layer 4 Switching | 50 |
# Configuring WMT Streaming Media Services on Standalone Content Engines

## Overview of the Windows Media Services Streaming Solution

## Overview of the WMT Streaming and Caching Services

### About the WMT RTSP Protocol

- How Standalone Content Engines Process WMT Requests
- About WMT Streaming and Caching Services with Standalone Content Engines
- About Streaming Policies in WMT Streaming Media Caching
- About WMT Proxy Caching
- About WMT Transparent Caching
- About Live Splitting with WMT
- About Proxy Authentication for a WMT-Enabled Content Engine

## Configuration Guidelines

- WMT Proxy Server Requirements
- Checklist for Configuring WMT Streaming and Caching Services on Standalone Content Engines

## Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines

- Checklist for Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines

## Enabling WMT Licenses on Standalone Content Engines

## Configuring General WMT Settings on Standalone Content Engines

- Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates
- About Variable WMT Bit Rates
- Configuring Subnet-Based Outgoing Bandwidth
- Configuring a WMT Bandwidth Incoming Bypass List
- Configuring Fast Streaming Features on Standalone Content Engines
- Configuring Fast Start on Standalone Content Engines
- Configuring Fast Cache on Standalone Content Engines

## Configuring Transparent Redirection of WMT Requests

- Configuring RTSP Transparent Redirection of WMT Requests

## Enabling and Configuring WMT Caching on Standalone Content Engines

- Enabling and Configuring Nontransparent WMT Proxy Caching on Standalone Content Engines
- Enabling and Configuring WMT Transparent Caching on Standalone Content Engines

## Configuring Standalone Content Engines to Distribute VOD Files

- Verifying That Preloaded VOD Files Are Cached and Properly Distributed to Windows Media Clients

## Configuring Standalone Content Engines to Deliver WMT Live Streams

- Configuring Standalone Content Engines to Multicast Live WMT Streams
  - Configuring Multicast-In Multicast-Out on Standalone Content Engines
  - Configuring Unicast-In Multicast-Out on Standalone Content Engines
  - Defining WMT Multicast Stations and Multicast Schedules on Standalone Content Engines
Contents

11

1

Cisco ACNS Software Configuration Guide for Locally Managed Deployments

OL-9140-01

1

PART 3

CHAPTER 10

Configuration of Content Services for Standalone Content Engines

Configuring Content Authentication and Authorization on Standalone Content Engines

About Authentication and Authorization of Content Requests

About End-to-End and Pass-through Authentication

About NTLM Support on Content Engines

About NTLM Support

Configuring Pass-Through Authentication for WMT Requests

Configuring End-to-End Authentication for HTTP Requests

Basic End-to-End Authentication

NTLM End-to-End Authentication

Configuring Request Authentication for HTTP Requests

Controlling Content Access for HTTP Requests

Authenticating Web Clients Through Pass-Through Authentication

Authenticating Web Clients Through HTTP Request Authentication

Guidelines for Configuring Request Authentication for HTTP Requests

Configuring an Authentication Service on Standalone Content Engines

Configuring the RADIUS Authentication Service

OL-9140-01

11
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Configuring Content Preloading and URL Filtering on Standalone Content Engines</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Configuring Content Preloading for Standalone Content Engines</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Support of Preloading of NTLM Authenticated Objects</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Creating a Preload URL List</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Enabling and Configuring Content Preloading on Standalone Content Engines</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Proxy Authentication Support for Content Preloading</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Example of Configuring Proxy Authentication for Content Preloading</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Stopping or Resuming Content Preloading on Standalone Content Engines</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Configuring URL Filtering on Standalone Content Engines</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Configuring Local List URL Filtering on Standalone Content Engines</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Example of Configuring URL Filtering with Local URL Lists</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Reloading Local List Files on Standalone Content Engines</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Creating Custom Blocking Messages</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Configuring Standalone Content Engines for N2H2 URL Filtering</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Configuring Standalone Content Engines for Websense URL Filtering</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>About Websense Server Failover</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>About Websense Services</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Configuring Ports for the Websense Server</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Websense Issues When Upgrading to ACNS 5.4.x Software</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Configuring the TACACS+ Authentication Service</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Configuring the LDAP Authentication Service</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Example of Configuring LDAP Authentication Service</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Understanding the Structure of the LDAP Database</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Configuring the LDAP Cache on Standalone Content Engines</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Querying LDAP Servers for User Membership Information</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Configuring the NTLM Authentication Service</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>About NTLM Load Balancing for HTTP Request Authentication</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Configuring the Content Engine to Use NTLM Servers for HTTP Request Authentication</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Configuring a List of Allowed Domains List NTLM HTTP Request Authentication</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Configuring the Authentication Method Control for NTLM HTTP Request Authentication</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Displaying the Current NTLM Configuration for Standalone Content Engines</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Configuring Group-Based Authorization for HTTP Requests</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Configuring Group-Based Access Lists on Standalone Content Engines</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Configuring Content Engines for Active Directory Group Searches</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>Disabling Group Name-Based Access Lists</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>Configuring the LDAP Acceptable Use Policy Feature</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>Configuring the LDAP Password Expiration Feature</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>Configuring Request Authentication for Nontransparent FTP Native Requests</td>
<td>55</td>
</tr>
</tbody>
</table>
Contents

13 Cisco ACNS Software Configuration Guide for Locally Managed Deployments

OL-9140-01

Websense Issues When Downgrading to an Earlier Version of the ACNS Software 29
Example of Configuring Websense Server Failover and URL Filtering 30
Configuring URL Filtering with a Local Websense Server 33
Configuring Websense URL Filtering with External Websense Servers 36
Deactivating Local Websense Server Services on Standalone Content Engines 37
Saving Websense Configuration Files 38
Viewing Websense URL Filtering Statistics 38
Configuring URL Filtering with SmartFilter Software 39
  About the SmartFilter Control List 40
  About the Temporary User Override Feature 40
Configuring Content Engines to Bypass URL Filtering for Specific HTTP and HTTPS Requests 41
Displaying the Current URL Filtering Configurations 41
Displaying URL Filtering Statistics 41
Clearing URL Filtering Statistics 42

CHAPTER 12

Configuring ICAP on Standalone Content Engines 1

Overview of ICAP 1
  About ICAP Services 2
    About ICAP Services and Vectoring Points 2
    About ICAP Performance 3
  ACNS Software and ICAP Services Interoperability Notes 3
Example of Configuring ICAP Services on a Content Engine 4
Configuring ICAP for Standalone Content Engines 5
  Configuring ICAP Settings for Standalone Content Engines 5
  Configuring ICAP Services on Standalone Content Engines 7
  Configuring an ICAP Server for Standalone Content Engines 8
Configuring Logging of ICAP Exchanges 9
Displaying Information About an ICAP Configuration 10
Displaying Statistics for ICAP Services 10

CHAPTER 13

Configuring the Rules Template on Standalone Content Engines 1

Rules Template Overview 1
  Supported Rule Actions per Protocol 3
Understanding Actions and Patterns 5
  Supported Rule Patterns 7
  Supported Rule Actions 11
    Example of no-url-filtering Action 14
    Example of use-proxy Action 14
### Contents

- Example of use-server Action 16
- Example of no-proxy Action 16
- Supported Action and Pattern Combinations 17
- Rules Template Processing Considerations 18
  - Rule Command for Converting Hostnames to IP Addresses 19
- Execution Order of Rule Actions 20
- Execution Order of Rule Patterns 21

- Configuring the Rules Template 22
- Enabling Rules Processing 23
- Configuring Rules for Standalone Content Engines 23
  - Configuring a Pattern List 28
  - Adding a Pattern to an Existing Pattern List 29
  - Associating an Action with an Existing Pattern List 30
  - Verifying an Action Performed on a Pattern List 30
  - Examples of Configuring Rules for Standalone Content Engines 31

- Displaying Statistics for Configured Rules 33
- Clearing Statistics for Configured Rules 33
- Using Regular Expressions for Rules 34
  - Special Characters in Regular Expressions 34
- Rules Caveats and Performance Guidelines 35
  - Pattern List Group-Type Caveats 35
  - Rules Configuration Guidelines for Best Performance 35

---

**PART 4**

**CHAPTER 14**

**Advanced Configuration of Standalone Content Engines**

**Configuring Primary and Backup Proxy Servers for Standalone Content Engines** 1

- Configuring Primary Proxy Failover 2
- Designating a Primary Outgoing HTTP Proxy Server 3
- Designating a Primary Outgoing FTP Proxy Server 3
- Designating a Primary Outgoing HTTPS Proxy Server 4
- Configuring HTTP and HTTPS Outgoing Proxy Exclusion Settings 5
- Monitoring Outgoing Proxy Servers and Statistics 7
- Displaying the Current Outgoing Proxy Server Configuration 8
- Displaying Outgoing Proxy Server Statistics 8

**CHAPTER 15**

**Configuring Advanced Transparent Caching Features on Standalone Content Engines** 1

- About Advanced Transparent Caching Features 2
- Configuring Bypass Settings on Standalone Content Engines 3
Chapter 16: Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines

Configuring Additional Network Interfaces 2
  About Virtual Interfaces 2
  Configuring Port Channels (EtherChannel) 2
  Configuring Standby Groups 4
  Specifying the Primary Interface 8
  Configuring Multiple Secondary IP Addresses on a Single Physical Interface 9
  Configuring the Fibre Channel Interface 9
  Defining Interface Descriptions 10

Configuring Bandwidth for Interfaces and Content Services 11
  Configuring Interface Bandwidth 11
  Configuring Bandwidth for Content Services 12

Chapter 17: Configuring Administrative Login Authentication and Authorization on Standalone Content Engines

Understanding Administrative Login Authentication and Authorization 2
  Default Administrative Login Authentication and Authorization Configuration 3
  Understanding Failover for Administrative Login Authentication 4
  Understanding Login Authentication and Authorization Through the Local Database 6
  Understanding RADIUS Authentication and Authorization 6
  Understanding TACACS+ Authentication and Authorization 7
  TACACS+ Enable Password Attribute 8

Configuring Administrative Login Authentication and Authorization 8
  Specifying RADIUS Authentication Settings for Standalone Content Engines 10
  Specifying TACACS+ Authentication Settings for Standalone Content Engines 12
  Specifying and Enabling the Administrative Login Authentication and Authorization Scheme 14
  Usage Guidelines 14
Contents

16
Cisco ACNS Software Configuration Guide for Locally Managed Deployments

16
Reenabling and Disabling Administrative Login Authentication and Authorization Through the Local Database

17
Enabling and Disabling Administrative Login Authentication and Authorization Through RADIUS

18
Enabling and Disabling Administrative Login Authentication and Authorization Through TACACS+

Displaying the Current Administrative Authentication and Authorization Configuration

CHAPTER 18
Configuring AAA Accounting on Standalone Content Engines

1
About AAA Accounting

2
About System Accounting

3
About EXEC Shell Accounting

3
About Command Accounting

4
Configuring AAA Accounting Settings on Standalone Content Engines

5
Displaying the AAA Accounting Configuration for Standalone Content Engines

8
Displaying and Clearing AAA Accounting Statistics on Standalone Content Engines

8

CHAPTER 19
Creating and Managing IP Access Control Lists for Standalone Content Engines

1
Introducing IP ACLs for Standalone Content Engines

2
Implementing IP ACLs for Standalone Content Engines

3
Example of Defining and Activating IP ACLs

4
Understanding the Basics About Working with IP ACLs

5
Working with Standard IP ACLs

5
Accessing Standard IP ACL Configuration Mode

5
Working with Extended IP ACLs

5
Accessing Extended IP ACL Configuration Mode

6
Defining and Activating IP ACLs on Standalone Content Engines

6
Usage Guidelines

6
Usage Guidelines for IP ACL Configuration Modes

9
Usage Guidelines for IP ACL Names

9
Creating or Modifying IP ACLs on Standalone Content Engines

10
IP ACL Conditions

14
Activating an IP ACL on an Interface

16
Applying an IP ACL to an Application

17
Using IP ACLs to Control SNMP Access

18
Using IP ACLs to Control TFTP Access

18
Using IP ACLs to Control Native FTP Access

19
Using IP ACLs to Control WCCP Access

20
Configuring WCCP Access Lists for Standalone Content Engines

20
CHAPTER 20
Viewing and Modifying TCP Stack Parameters on Standalone Content Engines

TCP Stack Overview
Viewing or Modifying TCP Parameters on Standalone Content Engines
Configuring TCP Memory Limits
Displaying TCP Configuration Information
TCP-Over-Satellite Extensions

CHAPTER 21
Monitoring and Troubleshooting

Monitoring Standalone Content Engines and Transactions
Monitoring Standalone Content Engines
Monitoring Standalone Content Engines with the Cisco Discovery Protocol
Monitoring Standalone Content Engines with SNMP
  Understanding the Different Versions of SNMP
  Supported MIBs
  Downloading MIB Files to Standalone Content Engines
  Enabling the SNMP Agent on Standalone Content Engines
  Defining SNMP Users for Standalone Content Engines
  Configuring Standalone Content Engines to Send SNMP Traps
  Disabling the SNMP Agent on Standalone Content Engines
  Disabling SNMP Traps on Standalone Content Engines
  Monitoring Standalone Content Engines with the ACNS Software Alarms
    Alarm Severity Levels
    Alarm Overload
    Checking for Application Liveness
    Configuring SNMIP Alarm Traps on Standalone Content Engines
    Displaying Information About ACNS Software Alarms
    Displaying Details About ACNS Software Alarms
    Displaying the History of ACNS Software Alarms
    Displaying the Status of ACNS Software Alarms
  Monitoring Critical Disk Drives on Standalone Content Engines
  Specifying the Disk Error-Handling Threshold
  Manually Marking and Unmarking Content Engine Disk Drives
## Contents

- Proactively Monitoring Disk Health with SMART 20
- System Logging with Standalone Content Engines 21
  - Displaying the Current Configuration for System Logging 22
  - Configuring System Logging on Standalone Content Engines 22
  - Configuring System Logging to the Console 23
  - Configuring System Logging to Disk 24
  - Configuring System Logging to Remote Syslog Hosts 24
- Mapping Syslog Priority Levels to RealProxy Error Codes 26
- Using CiscoWorks2000 26
- Monitoring Transactions with Standalone Content Engines 27
  - Displaying Statistics for Particular Protocols 28
  - Using ACNS Software Transaction Logs 31
- Enabling Transaction Logging 33
  - Logging FTP Client Usernames 34
  - Enabling Squid-Style Transaction Logging 34
  - Enabling Extended Squid-Style Transaction Logging 35
  - Enabling Apache-Style Transaction Logging 36
  - Enabling Custom Format Transaction Logging 36
- Logging Windows Domain with Authenticated Usernames 42
- Sanitizing Transaction Logs 42
- Exporting Transaction Log Files 43
  - Exporting Transaction Logs to External FTP or STFP Servers 43
- Changing the Transaction Logging Export Settings on Standalone Content Engines 45
- Restarting Export After Receiving a Permanent Error from the External FTP Server 45
- Archiving the Working Log 46
- Disabling Transaction Logging Export on Standalone Content Engines 47
- Monitoring HTTP Request Authentication Failures in Real Time 48
- Configuring the Remote Syslog Host for Real-Time Transaction Logging 49
- Configuring a Transaction Log Facility when Logging to a Remote Syslog Host 49
- Specifying the Transaction Log Entry Type when Logging to a Remote Syslog Host 50
- Displaying the Transaction Log Configuration for Standalone Content Engines 51
- Monitoring the Performance of Specific URLs 52

## Chapter 22

### Troubleshooting 1

- Overview 1
- Troubleshooting with HTTP Statistics Tools 2
  - Increasing the Cache Hit Rate 4
- Troubleshooting with the Debug Tool 4
- Troubleshooting URL Performance 5
Troubleshooting with the HTTP CLI Client 7
Troubleshooting in Real Time 9
Troubleshooting with the Log Viewing Tool 9
Troubleshooting with the Telnet Client 10
Troubleshooting with Ping 10
Troubleshooting with Traceroute 10
Troubleshooting with the WMT Diagnostic Tools 11

PART 6
Reference Material for Standalone Content Engines

APPENDIX A
Content Engine GUI Menu Options 1
Content Engine GUI Tabs 2
WCCP Tab and Subtabs 2
Caching Tab and Subtabs 4
System Tab and Subtabs 6
Reporting Tab and Subtabs 7

APPENDIX B
Reference Material for Standalone Content Engine Deployments 1
Supported Network Protocols 2
Supported Streaming Media Protocols 3
Supported WCCP Services 3
Matrix of Supported Caching, Filtering, and Authentication Methods 7
Supported Access Control and Filtering Services for Content Requests 7
ACNS Software CLI Command Modes for Standalone Content Engines 8
ACNS Software CLI Online Help and Keyboard Shortcuts 10
Unusable Multicast Address Assignments 11

INDEX
Preface

This preface describes who should read the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*, how it is organized, and its document conventions. This preface contains the following sections:

- **Document Objectives**, page 21
- **Audience**, page 22
- **Document Organization**, page 23
- **Document Conventions**, page 26
- **Related Documentation**, page 27
- **Obtaining Documentation**, page 27
- **Documentation Feedback**, page 28
- **Cisco Product Security Overview**, page 28
- **Obtaining Technical Assistance**, page 29
- **Obtaining Additional Publications and Information**, page 31

**Document Objectives**

This guide is intended for administrators who want to configure, manage, and monitor standalone Content Engines that are running Cisco Application and Content Networking System (ACNS) 5.5 software.

The term *standalone Content Engines* is used throughout this guide to refer to Content Engines that the ACNS administrators have intentionally not registered with a Content Distribution Manager so that they can configure, manage, and monitor these Content Engines as standalone devices.

The term *locally managed deployments* is used throughout this guide to refer to deployments that consist of one or more standalone Content Engines that are running the ACNS 5.x software and are configured as caching and streaming engines.
Note
To initially configure a Content Engine as a standalone device, you turn off the autoregistration feature so that the Content Engine will not automatically register with the Content Distribution Manager, and so that you can individually manage it through the Content Engine command-line interface (CLI) or the Content Engine graphical user interface (GUI) as a standalone device.

The Content Engine GUI allows you to configure, manage, and monitor standalone Content Engines remotely through your browser, a console connection or a terminal emulation program. Although either the Content Engine GUI or the CLI can be used to configure and manage standalone Content Engines, the instructions and examples in this guide primarily use the CLI method. (Certain features can be configured through the Content Engine CLI only.) The Content Engine GUI has context-sensitive online help that can be accessed by clicking the HELP button. See Appendix A, “Content Engine GUI Menu Options,” for a complete list of Content Engine GUI options.

This guide explains how to configure, manage, and monitor standalone Content Engines running the ACNS 5.4 software for the following purposes:

- Transparent forward caching deployments
  - For conventional caching (DNS, HTTP, HTTPS, and native FTP caching)
  - For Windows Media Technologies (WMT) transparent caching
  - For RealMedia transparent caching
- Transparent reverse proxy caching deployments (HTTP caching for reverse proxy packets)
- Nontransparent forward proxy caching deployments:
  - For conventional caching (HTTP, HTTPS, and FTP-over-HTTP caching)
  - For WMT proxy caching
  - For RealMedia proxy caching
- WMT streaming deployments
- Real-Time Streaming Protocol (RTSP) streaming deployments

Note
If you are using content routing, you must use the Content Distribution Manager. For information about configuring a centrally managed ACNS network device (Content Engines or Content Routers that are registered with a Content Distribution Manager), see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

Audience
This guide is intended for administrators who want to configure, manage, and monitor standalone Content Engines. The administrator should be familiar with Cisco router and switch configuration. An understanding of caching and streaming concepts is necessary. This guide is not a tutorial.
Document Organization

This guide includes the following chapters and appendixes that are divided into six parts:

- Overview that introduces some basic concepts and the typical ways to deploy standalone Content Engines
- Basic configuration for standalone Content Engines
- Configuration of content services for standalone Content Engines
- Advanced configuration of standalone Content Engines
- Monitoring and troubleshooting of standalone Content Engines
- Reference material that is pertinent to configuring and monitoring standalone Content Engines (for example, a list of Content Engine GUI options, a list of supported WCCP services, and a matrix of supported caching, filtering, and authentication mechanisms per protocol)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>Overview</td>
<td>Provides a brief overview of the ACNS network solution, and introduces the typical ways to deploy a standalone Content Engine.</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>Introduction</td>
<td>Provides an overview of some basic concepts that are important to understand before you configure a standalone Content Engine for caching and streaming.</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Deployment Scenarios for Standalone Content Engines</td>
<td>Describes the typical ways that you can deploy standalone Content Engines as streaming and caching engines.</td>
</tr>
<tr>
<td>Part 2</td>
<td>Basic Configuration for Standalone Content Engines</td>
<td>Describes the procedures for configuring a basic configuration on standalone Content Engines. Includes instructions on how to use the interactive Setup utility to configure a basic configuration (device network settings, disk configurations, and some commonly used caching services) on standalone Content Engines. Also provides some important information about how to get started (for example, how to log in to a standalone Content Engine and preload content on it).</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Performing Other Basic Tasks for Standalone Content Engines</td>
<td>Describes how you can use the Content Engine CLI to perform other basic tasks such as setting the system clock and managing login accounts.</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Configuring Transparent Redirection for Standalone Content Engines</td>
<td>Describes how to configure WCCP and Layer 4 switching as redirection methods that transparently intercept and redirect content requests (caching and streaming) to standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Configuring Conventional Caching Services for Standalone Content Engines</td>
<td>Describes how to configure conventional caching services (DNS, HTTP, FTP, and HTTPS caching) on standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Configuring RealMedia Services on Standalone Content Engines</td>
<td>Describes how to configure RealMedia streaming and caching services on standalone Content Engines. Also describes how to configure the Real-Time Streaming Protocol (RTSP) gateway, which runs on the Content Engine, and directs RTSP requests to the appropriate backend RTSP server (for example, the RealProxy server).</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Configuring WMT Streaming Media Services on Standalone Content Engines</td>
<td>Describes how to configure Windows Media Technologies (WMT) streaming and caching services on standalone Content Engines.</td>
</tr>
<tr>
<td>Part 3</td>
<td>Configuration of Content Services for Standalone Content Engines</td>
<td></td>
</tr>
<tr>
<td>Chapter 10</td>
<td>Configuring Content Authentication and Authorization on Standalone Content Engines</td>
<td>Describes how to configure access control on a Content Engine for processing HTTP, HTTPS, and FTP requests for content.</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Configuring Content Preloading and URL Filtering on Standalone Content Engines</td>
<td>Describes how to configure a standalone Content Engine to configure content preloading and URL filtering on standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Configuring ICAP on Standalone Content Engines</td>
<td>Describes how to configure a standalone Content Engine (HTTP proxy server) to use the Internet Content Adaptation Protocol (ICAP) to communicate with an external ICAP server that filters and adapts the requested content.</td>
</tr>
<tr>
<td>Chapter 13</td>
<td>Configuring the Rules Template on Standalone Content Engines</td>
<td>Describes how to configure a standalone Content Engine to use a set of configured rules to filter HTTP, HTTPS, FTP-over-HTTP, WMT, and RTSP requests. These configured rules rewrite certain headers, redirect the request, or otherwise manipulate the request.</td>
</tr>
</tbody>
</table>

| Part 4 | Advanced Configuration of Standalone Content Engines | Describes how to configure primary and backup (failover) proxy servers for standalone Content Engines. |

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Chapter 15</td>
<td>Configuring Advanced Transparent Caching Features on Standalone Content Engines</td>
<td>Describes how to configure advanced transparent caching features (for example, IP spoofing, traffic bypass, and flow protection) on standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 16</td>
<td>Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines</td>
<td>Describes how to set up additional network interfaces and configure bandwidth for these interfaces and content services on standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 17</td>
<td>Configuring Administrative Login Authentication and Authorization on Standalone Content Engines</td>
<td>Describes how to configure a standalone Content Engine to use specific login authentication mechanisms (local, RADIUS, or TACACS+) to process administrative login requests (requests from administrators who want to log on to a standalone Content Engine for configuration, monitoring, or troubleshooting purposes).</td>
</tr>
<tr>
<td>Chapter 18</td>
<td>Configuring AAA Accounting on Standalone Content Engines</td>
<td>Describes how to configure authentication, authorization, and accounting (AAA) accounting using TACACS+ for standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 19</td>
<td>Creating and Managing IP Access Control Lists for Standalone Content Engines</td>
<td>Describes how to configure and manage IP access control lists (ACLs) to control access to specific applications or interfaces on standalone Content Engines.</td>
</tr>
<tr>
<td>Chapter 20</td>
<td>Viewing and Modifying TCP Stack Parameters on Standalone Content Engines</td>
<td>Describes how to view or modify TCP stack parameters on standalone Content Engines.</td>
</tr>
<tr>
<td>Part 5</td>
<td>Monitoring and Troubleshooting</td>
<td></td>
</tr>
<tr>
<td>Chapter 21</td>
<td>Monitoring Standalone Content Engines and Transactions</td>
<td>Describes how to monitor standalone Content Engines and transactions.</td>
</tr>
<tr>
<td>Chapter 22</td>
<td>Troubleshooting</td>
<td>Describes troubleshooting with standalone Content Engines.</td>
</tr>
<tr>
<td>Part 6</td>
<td>Reference Material for Standalone Content Engines</td>
<td></td>
</tr>
<tr>
<td>Appendix A</td>
<td>Content Engine GUI Menu Options</td>
<td>Describes the tabs and subtabs (menu options) that are available from the Content Engine GUI. This GUI is an alternative method to the Content Engine CLI for configuring and monitoring standalone Content Engines.</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Reference Material for Standalone Content Engine Deployments</td>
<td>Contains important reference material (for example, a list of supported WCCP services and a matrix of supported caching, filtering, and authentication mechanisms per protocol) that is pertinent to configuring and monitoring standalone Content Engines.</td>
</tr>
</tbody>
</table>
## Document Conventions

This document uses the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong> font</td>
<td>Commands, keywords, and button names are in <strong>boldface</strong>.</td>
</tr>
<tr>
<td>italic font</td>
<td>Variables for which you supply values are in <em>italics</em>. Directory names and file names are also in italics.</td>
</tr>
<tr>
<td>screen font</td>
<td>Terminal sessions and information the system displays are printed in <strong>screen font</strong>.</td>
</tr>
<tr>
<td><strong>boldface</strong> screen font</td>
<td>Information you must enter is in <strong>boldface</strong> <strong>screen</strong> font.</td>
</tr>
<tr>
<td><em>italic</em> screen font</td>
<td>Variables you enter are printed in <em>italic</em> <strong>screen</strong> font.</td>
</tr>
<tr>
<td>plain font</td>
<td>Enter one of a range of options as listed in the syntax description.</td>
</tr>
<tr>
<td>^D or Ctrl-D</td>
<td>Hold the Ctrl key while you press the D key.</td>
</tr>
<tr>
<td>string</td>
<td>Defined as a nonquoted set of characters.</td>
</tr>
<tr>
<td></td>
<td>For example, when setting a community string for SNMP to “public,” do not use quotation marks around the string, or the string will include the quotation marks.</td>
</tr>
<tr>
<td>Vertical bars (</td>
<td>Vertical bars separate alternative, mutually exclusive, elements.</td>
</tr>
<tr>
<td>{ }</td>
<td>Elements in braces are required elements.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Elements in square brackets are optional.</td>
</tr>
<tr>
<td>{x</td>
<td>y</td>
</tr>
<tr>
<td>[x</td>
<td>y</td>
</tr>
<tr>
<td>[{} ]</td>
<td>Braces within square brackets indicate a required choice within an optional element.</td>
</tr>
</tbody>
</table>

---

**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 ACNS software, see the following documentation:

- Documentation Guide and License and Warranty for Cisco ACNS Software, Release 5.5
- Cisco ACNS Software Upgrade and Maintenance Guide, Release 5.x
- Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5
- Cisco ACNS Software Command Reference, Release 5.5
- Cisco ACNS Software API Guide, Release 5.5
- Release Notes for Cisco ACNS Software, Release 5.5

Obtaining Documentation

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

Cisco.com

You can access the most current Cisco documentation at this URL:
http://www.cisco.com/techsupport

You can access the Cisco website at this URL:
http://www.cisco.com

You can access international Cisco websites at this URL:

Product Documentation DVD

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

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

Registered Cisco.com users may order Cisco documentation at the Product Documentation Store in the Cisco Marketplace at this URL:

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

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

Documentation Feedback

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

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

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

We appreciate your comments.

Cisco Product Security Overview

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


From this site, you will find information about how to:

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

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

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

To see security advisories, security notices, and security responses as they are updated in real time, you can subscribe to the Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed. Information about how to subscribe to the PSIRT RSS feed is found at this URL:

Reporting Security Problems in Cisco Products

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

- For Emergencies only—security-alert@cisco.com
  
  An emergency is either a condition in which a system is under active attack or a condition for which a severe and urgent security vulnerability should be reported. All other conditions are considered nonemergencies.

- For Nonemergencies—psirt@cisco.com

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

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

Tip

We encourage you to use Pretty Good Privacy (PGP) or a compatible product (for example, GnuPG) to encrypt any sensitive information that you send to Cisco. PSIRT can work with information that has been encrypted with PGP versions 2.x through 9.x.

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


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

If you do not have or use PGP, contact PSIRT at the aforementioned e-mail addresses or phone numbers before sending any sensitive material to find other means of encrypting the data.

Obtaining Technical Assistance

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

Cisco Technical Support & Documentation Website

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

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


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

Submit a Service Request

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

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

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

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

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

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

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

Definitions of Service Request Severity

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

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

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operations are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.
Severity 3 (S3)—Operational performance of the network is impaired, while most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

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

Obtaining Additional Publications and Information

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

- The *Cisco Product Quick Reference Guide* is a handy, compact reference tool that includes brief product overviews, key features, sample part numbers, and abbreviated technical specifications for many Cisco products that are sold through channel partners. It is updated twice a year and includes the latest Cisco offerings. To order and find out more about the Cisco Product Quick Reference Guide, go to this URL: http://www.cisco.com/go/guide

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

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

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

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

- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL: http://www.cisco.com/ipj
- Networking products offered by Cisco Systems, as well as customer support services, can be obtained at this URL:
- Networking Professionals Connection is an interactive website for networking professionals to share questions, suggestions, and information about networking products and technologies with Cisco experts and other networking professionals. Join a discussion at this URL:
  http://www.cisco.com/discuss/networking
- World-class networking training is available from Cisco. You can view current offerings at this URL:
Introduction

This chapter provides an overview of the Cisco Application and Content Networking System (ACNS) solution. This chapter contains the following sections:

- Overview of the ACNS Software Solution, page 1-1
- Overview of Standalone Content Engine Capabilities, page 1-6

The term standalone Content Engines is used throughout this guide to refer to Content Engines that ACNS administrators have intentionally not registered with a Content Distribution Manager (if there is one in the network) so that they can configure, manage, and monitor these Content Engines as standalone devices. This guide focuses solely on deploying, managing, and monitoring standalone Content Engines that are running the ACNS 5.5 software. Multiple standalone Content Engines can be deployed (for example, you can deploy clusters of standalone Content Engines). You can configure standalone Content Engines through the Content Engine CLI, Content Engine GUI, or the Setup utility that was introduced in the ACNS 5.2.1 software release.

For information about how to deploy, manage, and monitor Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

Overview of the ACNS Software Solution

With the advent of e-business applications such as e-commerce, e-learning, knowledge sharing, and corporate communications, networks can experience uncontrollable bottlenecks in the flow of traffic. The ACNS solution helps enterprises and Internet service providers (ISPs) protect their networks from these uncontrollable bottlenecks and efficiently distribute rich media files to their end users. Designed for affordability and ease of installation, the ACNS solution enables high-impact, high-bandwidth rich media such as high-quality streaming video to be quickly deployed with minimal administration.

Streaming is a technology that allows content to be accessed or viewed before all the media packets have been received, with caching, the content must be received in its entirety before it can be accessed.

ACNS software supports multiple content routing methods and allows Content Engine caches to be populated with content in multiple ways. Content Engines with ACNS 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.
By caching content locally, 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 over IP (VoIP).

For example, enterprises can deploy one or more Content Engines at each branch office, configure access control and filtering on each of these Content Engines, and then push content to these Content Engines. The Content Engine at each branch office uses specific policies to determine if a content request should be denied or granted. If access to the content is granted, the Content Engine checks its local cache for a copy of the content. If the content is already stored in its local cache, the Content Engine sends the client the cached content; otherwise, it retrieves the content from the source (origin server), caches the content, and sends the cached content to the client. When the Content Engine receives subsequent client requests for the same content, it sends the client the cached content instead of retrieving the same content from the origin server again. See Figure 1-1 for a sample deployment.

**Figure 1-1  Sample Enterprise Deployment for Standalone Content Engines at Branch Offices**

Content Engines (transparent forward proxy server)

Origin servers (content sources)

Internet

TCP connection

WCCP

Branch office (network edge)

1. Client browser performs DNS lookup to resolve IP address of the specified content source

2. Web client sends content request to source

3. Content request to source is transparently intercepted and redirected to Content Engine

4a. If cache hit, then 5

4b. If cache miss, Content Engine retrieves content from source, caches it, and sends to client (5)

5. Requested content sent to web client

Content Engine (transparent forward proxy server)

In this sample deployment, client requests are transparently redirected to the Content Engine at the branch office by a router that is running Cisco’s Web Cache Communication Protocol (WCCP). Other possible routing methods include transparent redirection through a Layer 4 Cisco Content Services Switch [CSS] switch) or direct proxy routing (web clients are explicitly configured to send their requests directly to the Content Engine).
The Setup utility expedites the process of getting standalone Content Engines up and running with a set of commonly used caching services (listed in Table 4-2).

### Types of ACNS Network Devices

Table 1-1 shows the three different types of ACNS network devices.

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Distribution Manager</td>
<td>Is a centralized content and device management station that includes the configuration and monitoring of ACNS 5.x network devices (Content Routers, Content Engines that are registered with the Content Distribution Manager), content acquisition and distribution, and services. Allows you to centrally manage devices (Content Engines and Content Routers) as groups instead of individually.</td>
<td>See the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.</td>
</tr>
</tbody>
</table>
| Content Engine             | Serves the requested content to the client. Content Engines can be deployed in one of two ways:  
• Standalone Content Engines (the focus of this guide)  
or  
• Content Engines that are registered with a Content Distribution Manager (see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5). | For an overview of standalone Content Engines, see the “About Standalone Content Engines” section on page 1-4. |
| Content Router             | Redirects content requests to the registered Content Engine that is closest to the client. This type of request redirection is referred to as content routing. With content routing, a Content Engine must be registered with a Content Distribution Manager. Standalone Content Engines do not support content routing. | See the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5. |

Note: The ACNS software device mode determines whether the ACNS device is functioning as a Content Distribution Manager, Content Engine, or Content Router. You can specify the device mode through the **device mode** global configuration command. The default device mode is Content Engine.
As the following sample deployments indicate, all three types of ACNS network devices do not need to be present in order for the ACNS 5.x network to function:

- **Deployment A**—A single standalone Content Engine (no Content Distribution Manager or Content Router)
- **Deployment B**—Several standalone Content Engines (no Content Distribution Manager or Content Router)
- **Deployment C**—One Content Distribution Manager, several Content Engines that are registered with a Content Distribution Manager, and no Content Router
- **Deployment D**—One Content Distribution Manager, several Content Engines that are registered with a Content Distribution Manager, and one or more Content Routers

### About Standalone Content Engines

Content Engines accelerate any HTTP-deliverable and streaming media-type content by storing and delivering content close to the end users (web clients) on their local networks. The term *web client* denotes an end user who is using a browser or media player to request content or information. For more information about the supported web clients, see the “Web Clients Supported by Standalone Content Engines” section on page 1-12.

Standalone Content Engines can be deployed in one of the following ways:

- Inside an enterprise firewall on an internal network
- At the edge of the enterprise network (in branch offices)

**Note**

Multiple standalone Content Engines can be deployed (for example, you can deploy clusters of standalone Content Engines).

Standalone Content Engines can service content requests from the following types of clients:

- Web browsers (for example, Microsoft Internet Explorer)
- Streaming media players (for example, Windows Media players, RealMedia players [RealPlayer and RealOne players]).

Client requests for content can be routed to standalone Content Engines in one or more of the following ways:

- Direct proxy routing in which the client browsers and media players are configured to send their requests directly to the Content Engine, which acts as a nontransparent forward proxy server
- Transparent redirection (routers and switches transparently intercept web requests and send them to the Content Engine for inspection and manipulation)
  - Cisco WCCP routing
  - Layer 4 switching

With transparent redirection, the WCCP-enabled router or a Layer 4 switch transparently intercepts the client request and redirects it to the Content Engine instead of to the intended server. The Content Engine poses as the intended server, acknowledges the request, and establishes the connection with the client. The client believes that it has established a connection with the intended server even though it is actually using a connection with the Content Engine. For more information about transparent redirection, see the “Deploying Caching and Streaming Services in Transparent Mode” section on page 3-6.
One or more routing methods can be used in a single environment. For example, you can configure a standalone Content Engine to use direct proxy routing for HTTP or FTP-over-HTTP requests but use transparent redirection for Windows Media Technologies (WMT) requests. The routing method that is used to route content requests to the standalone Content Engine determines the types of content services that the Content Engine can support. For more information, see the “Caching and Streaming Services with Standalone Content Engines” section on page 1-9.

Note
Content routing is not supported on standalone Content Engines. If content routing is used, you must register the Content Engines with the Content Distribution Manager. For information about content routing, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

Types of Content Served in ACNS Networks

Content is the fundamental element of the ACNS network; it represents all the data that the ACNS network handles. Content can be a file or a media stream, and may be on-demand, preloaded, pre-positioned, or live. Content can be classified based on how the content is acquired, distributed, or served.

Table 1-2 describes the different types of content that can be served in an ACNS 5.x network.

Note
Pre-positioned content is only supported on Content Engines that are registered with a Content Distribution Manager. Although pre-positioned content is not supported on standalone Content Engines, preloaded content is supported on standalone Content Engines.

<table>
<thead>
<tr>
<th>Type of Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-demand</td>
<td>Content that is acquired, cached, and delivered because of a user request (client-triggered demand), as shown in Figure 1-1. This type of caching is called demand-pull caching. You can configure demand-pull caching on a standalone Content Engine when it is operating in transparent mode (Content Engine receives requests through transparent redirection) or nontransparent mode (Content Engine receives requests directly from the web client). Cached content that is retrieved through HTTP is stored in the cache file system (cfs) storage partition on the Content Engine. Cached content that is retrieved through the two streaming protocols (WMT and RTSP) is stored in the media file system (mediafs) storage partition on the Content Engine.</td>
</tr>
<tr>
<td>Preloaded</td>
<td>Content that is retrieved and stored on a standalone Content Engine because you scheduled a retrieval of specific content in anticipation of user requests for that content. The following types of content can be preloaded on a standalone Content Engine: HTTP URLs and FTP-over-HTTP URLs. All configured HTTP and FTP-over-HTTP apply to the preloaded objects. During the preload process, the standalone Content Engine scans web sites several link levels down for content, retrieves the specified content, and stores it locally. When the Content Engine receives future requests for this content, it serves the content from its local storage, which results in WAN bandwidth savings as well as accelerated delivery of the content to the web client. For more information about preloading content on standalone Content Engines, see the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2.</td>
</tr>
</tbody>
</table>
Overview of Standalone Content Engine Capabilities

The section provides an overview of standalone Content Engine deployments, and includes the following sections:

- **Proxying Services with Standalone Content Engines**, page 1-7
- **Caching and Streaming Services with Standalone Content Engines**, page 1-9
- **Filtering and Access Control with Standalone Content Engines**, page 1-15
- **Monitoring and Troubleshooting Features with Standalone Content Engines**, page 1-17

For sample deployment scenarios, see Chapter 3, “Deployment Scenarios for Standalone Content Engines.”

---

<table>
<thead>
<tr>
<th>Type of Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-positioned</td>
<td>Content that is retrieved and distributed through a network of Content Engines that are registered with a Content Distribution Manager because the ACNS network administrator has configured acquisition and distribution of content on these Content Engines in anticipation of user requests. Used as a means of distributing content to populate Content Engines in a centrally managed ACNS network environment. Bandwidth-intensive content objects, such as Java applets, Macromedia Flash animations, Shockwave programs, and other file formats can be managed and scheduled for distribution to Content Engines during off-peak hours. For information about managing pre-positioned content, see the <em>Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.</em></td>
</tr>
<tr>
<td>Live</td>
<td>Content stream (typically streaming media) that is being broadcast from an origin server. (This content is acquired as a live streaming broadcast from either a satellite or a terrestrial broadcasting source.) You configure the policies associated with obtaining the live multimedia stream, such as the program listing URL (playlist), the maximum bit rate, and so forth, as well as the distribution policies, such as priority, schedule, and maximum bandwidth. For information about configuring a standalone Content Engine to deliver WMT live streams, see the “Configuring Standalone Content Engines to Deliver WMT Live Streams” section on page 9-37.</td>
</tr>
</tbody>
</table>

The remainder of this chapter focuses on standalone Content Engine deployments only. The routing methods and content services that are supported by a Content Engine vary based on whether the Content Engine is a standalone Content Engine or is a Content Engine that is registered with a Content Distribution Manager. For information about Content Engines that are registered with a Content Distribution Manager, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.*
Proxying Services with Standalone Content Engines

Standalone Content Engines can be deployed as proxy servers to provide the following core proxying services:

- Forward proxy caching
- Reverse proxy caching

**Note**

A proxy server is an intermediary server that accepts requests for content from clients. If the proxy server already has a copy of the requested content in its local storage (cache), it services these requests from its own local storage; otherwise, it forwards these requests to the origin server or other proxy servers. A proxy server functions as both a client and a server. It acts as a server to the web client that is requesting the content, and acts as a client to the servers (for example, the origin server or other proxy servers [for example, the specified upstream proxy servers]) that it connects to. A proxy server is commonly called a proxy.

**Forward Proxy Caching**

With forward proxy caching, the standalone Content Engine acts as a proxy server for web clients. The Content Engine (forward proxy server) provides internal clients access to the Internet through a firewall.

If the client browsers and media players are explicitly configured to send their content requests to the Content Engine (forward proxy server), this is referred to as **direct proxy routing**. When direct proxy routing is used to direct client requests to the Content Engine, the Content Engine is operating in nontransparent mode; the clients are aware that their requests are being directed to the Content Engine. The Content Engine uses specific policies and rules to determine whether a client should be granted or denied access to the requested Internet content. This type of forward proxying service is usually provided as part of a larger Internet security solution in enterprise environments. By implementing this service, enterprises enable their end users (web clients) to go outside the firewall without compromising the integrity of the company’s private network.

The direct proxy routing method is the most straightforward routing method. This routing method is typically used when the user desktops are tightly controlled. Consequently, direct proxy routing is generally used in enterprise environments as opposed to service provider environments. See Figure 1-2 for a sample deployment of forward proxy caching when direct proxy routing is used to direct client requests to the Content Engine. See Table 1-5 for a list of supported caching and streaming services when direct proxy routing is used to route requests to the Content Engine. For more information about deploying caching and streaming services with direct proxy routing, see the “Deploying Caching and Streaming Services in Nontransparent Mode” section on page 3-3.
You can also deploy a Content Engine as a transparent forward proxy server that provides the same benefits as direct proxy routing but does not require any configuration changes to client desktops. In this type of deployment, clients are not aware that their content requests are being redirected to the Content Engine (transparent forward proxy server). The transparent redirection method can be implemented through a WCCP-enabled router or a Layer 4 switch.

Although the transparent redirection method demands an understanding of network topology and traffic patterns, organizations generally prefer to use this method because it does not require any configuration changes to the client desktops. However, there may be a legacy requirement for using direct proxy routing. There also may be cases in which organizations need to use direct proxy routing for a particular service (for example, HTTPS proxy caching) because they are not allowed to make the necessary configuration changes to the WCCP-enabled router or switch at the branch office. For more information, see the “Overview of Transparent Reverse Proxy Caching” section on page 3-10.

Reverse Proxy Caching

With reverse proxy caching, the standalone Content Engine acts as a proxy server for the servers in a web server farm, and the web clients are not aware that their HTTP requests are being transparently redirected to the Content Engine (transparent reverse proxy server). In such deployments, the Content Engine acts as a proxy for the servers in the web server farm.
Overview of Standalone Content Engine Capabilities

A major difference between reverse proxy caching and forward proxy caching is that with reverse proxy caching the Content Engine is configured to cache only specific content (for example, only content from the web servers in the web server farm), with forward proxy caching, the Content Engine is configured to cache content whenever possible.

Reverse proxy caching also enables a standalone Content Engine to provide external clients access to internal content (for example, content on a company’s intranet) through a firewall. Typically, this reverse proxy caching is used for secure web publishing. In a reverse proxy cache configuration, the proxy server is configured with an Internet-routable IP address. Web clients are directed to the proxy server based on DNS resolution of a domain name. To a web client, the reverse proxy server appears to be the origin web server.

Some significant advantages to deploying reverse proxy caching on standalone Content Engines are as follows:

- It provides an alternative to web server expansion by offloading the processing of static images from the server farm. By having the reverse proxy server transparently handle inbound requests for content, web traffic is significantly reduced.
- It provides a possible way of replicating content to geographically dispersed areas by deploying Content Engines in these areas.
- It does not require any client configuration changes (you do not need to configure the client browsers to point to the Content Engine that is functioning as the reverse proxy server).

Reverse proxy caching is a WCCP service (service 99). Consequently, both the router and the Content Engine must be configured to run the reverse proxy service. See Table 1-6 for a list of supported caching and streaming services when transparent redirection is used to direct client requests to the Content Engine. For more information about deploying the reverse proxy caching service, see the “Overview of Transparent Reverse Proxy Caching” section on page 3-10.

Caching and Streaming Services with Standalone Content Engines

The ACNS software supports several types of caching and streaming media services. The types of services that you can configure on a standalone Content Engine vary depending on how the content request is routed to the Content Engine.

Table 1-3 lists the streaming media solutions supported by the ACNS 5.x software, including Microsoft Windows Media Technologies (WMT), and Real-Time Streaming Protocol (RTSP) solutions from RealNetworks, Inc., Apple Computer, and Cisco Systems, Inc.
Overview of Standalone Content Engine Capabilities

Table 1-3  ACNS 5.x Software Streaming Media Solutions

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Description of Solution</th>
<th>Standalone Content Engines</th>
<th>Content Engines Registered with Content Distribution Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft WMT</td>
<td>Microsoft solution for streaming media. Windows Media Services 9 (WMS 9) supports the RTSP/RTP protocol (IETF standard RTSP protocol plus proprietary extensions). In the ACNS 5.3 software release, support for WMS 9-over-RTSP was added.</td>
<td>WMT proxy and server are installed on the Content Engine. Windows Media clients request streaming media content.</td>
<td>Same as standalone Content Engine.</td>
</tr>
<tr>
<td>RealNetworks RealMedia</td>
<td>RealNetworks, Inc. solution for streaming media that uses RealNetworks RTSP protocol (IETF standard RTSP protocol plus proprietary extensions).</td>
<td>RealProxy is the only RealMedia component that can be installed. RealProxy uses the RealNetworks RTSP protocol to serve content to RealMedia clients (RealPlayer and RealOne players).</td>
<td>Same as standalone Content Engines except that all of the RealMedia components (for example, RealProxy and RealSubscriber) can be installed on the Content Engine because it is registered with a Content Distribution Manager.</td>
</tr>
<tr>
<td>Apple QuickTime</td>
<td>Apple Computer solution for streaming media that uses the IETF standard RTSP protocol.</td>
<td>CLI-based live and rebroadcast programs can be played back from the QuickTime player.</td>
<td>Cisco Streaming Engine runs on the Content Engine and delivers QuickTime-compliant content (for example, MOV and MPEG-1 content) to QuickTime clients.</td>
</tr>
<tr>
<td>Cisco Streaming Engine</td>
<td>Cisco RTSP-based streaming media solution that uses the IETF standard RTSP protocol.</td>
<td>Cisco Streaming Engine can be enabled on a stand-alone Content Engine for live and video-on-demand (VOD) streaming services for local users.</td>
<td>Cisco Streaming Engine runs on the Content Engine and delivers pre-positioned content. Also used to support IP/TV integration for live and video-on-demand (VOD) streaming services for local users.</td>
</tr>
</tbody>
</table>

See Table 1-5 and Table 1-6 for a list of supported caching and streaming services when direct proxy routing or transparent redirection is used to direct client requests to a standalone Content Engine.

RTSP is a standard Internet streaming control protocol (RFC 2326). It is an application-level protocol that controls the delivery of streaming media data with real-time properties, such as video and audio, and has been widely adopted as a streaming media protocol in the caching and ACNS environments. Apple QuickTime, RealNetworks RealProxy, and the Cisco Streaming Engine are all backend RTSP servers that use RTSP as a streaming media protocol. On standalone Content Engines, only RealProxy is supported.

The RealNetworks RealProxy feature uses the RealNetworks RTSP protocol, which includes proprietary extensions to the standard IETF standard RTSP protocols. The RealProxy feature enables a standalone Content Engine to support RealMedia transparent and proxy caching. RealProxy also enables the Content Engine to stream cached VOD files to RealMedia players, and to support live splitting.
The RealProxy feature is licensed RealNetworks software. To enable this feature on a Content Engine, you must have a RealProxy license key, which is supplied on a certificate shipped with the Content Engine. If you are downloading the ACNS 5.x software, you can purchase a RealProxy license through the Cisco.com website. For more information, see the “Configuring RealMedia Services” section on page 8-9. For more information on RTSP streaming media services with standalone Content Engines, see Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”

You can use the Cisco Streaming Engine to pre-position RTSP-based content on Content Engines that are registered with a Content Distribution Manager. The Cisco Streaming Engine uses the standard RTSP protocol to serve QuickTime-compliant content (for example, MOV and MPEG-1 content) to clients.

Cisco IP/TV is a member of the Cisco content networking product family. Cisco IP/TV consists of IP/TV Program Manager, one or more IP/TV Broadcast Servers, and IP/TV Viewer or the QuickTime web plug-in. The central management platform for the IP/TV network, IP/TV Program Manager, offers a simple browser interface to schedule and set policies for live and rebroadcast events and VOD files, as well as recording capabilities. IP/TV Broadcast Servers offer real-time encoding, and delivery of live, scheduled, and on-demand video.

When the Cisco Streaming Engine is used in conjunction with the IP/TV 5.1 software and later releases, it provides both live and VOD streaming services to local users. The following two examples show how the Cisco IP/TV solution (the IP/TV Release 5.1 and later releases) can be used in conjunction with the ACNS 5.1 software and later releases, and Content Engines that are registered with a Content Distribution Manager:

- Record live events created with IP/TV and deliver the content using the ACNS 5.1 software and later releases.
- Extend the reach of IP/TV live or rebroadcast events to multicast islands through the ACNS 5.1 software and later releases. The term multicast islands refers to non-multicast-enabled networks or environments with a combination of a unicast WAN and a multicast LAN.

In the ACNS 5.1 software, a fourth device mode (Program Manager) was added for certain devices to support IP/TV integration into ACNS software networks (the ACNS 5.1 software and later releases). For example, you can configure the Content Engine CE-565 or CE-7305 model as an IP/TV Program Manager by specifying Program Manager as the device mode. When you launch the Setup utility on a device that supports device mode changes, you are asked to specify the device mode. To configure a device as a Program Manager through the Setup utility, specify PM as the device mode:

```
What is the mode of the device (CE/CR/CDM/PM) [CE]: PM
```

Cisco IP/TV and the Cisco Streaming Engine are not supported on standalone Content Engines. For information about deploying the Cisco Streaming Engine, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5. For more information about IP/TV, see the Cisco IP/TV 5.x product documentation.
**Supported Protocols for Caching and Streaming**

The interaction between a web client (browser or media player) and a web server uses the existing standard application-layer Internet protocols such as HTTP and RTSP. The Content Engine has to be able to serve web objects to the web client using all of these web access protocols.

Table B-1 describes the network protocols that a Content Engine running the ACNS 5.5.x software can use to serve content to the web client. Table B-2 lists the streaming media protocols, control channels, the corresponding data format, and transport types that can be used to deliver streaming media files with standalone Content Engines.

Support for HTTP, FTP, TFTP, HTTPS, and the IETF standard RTP/RTSP protocols is included as part of the ACNS software product (the ACNS 5.1 software and later releases). Support for the following two features require a separate license:

- The WMT feature, which offers Windows Media Services, requires a WMT license. For more information, see the “Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines” section on page 9-14.
- The RealNetworks RealProxy feature, which uses RealNetworks’ RTSP protocol that includes proprietary extensions to the standard IETF standard RTSP protocol, requires a RealProxy license. For more information, see the “Configuring RealMedia Services” section on page 8-9.

### Web Clients Supported by Standalone Content Engines

Table 1-4 lists the web clients that can communicate with standalone Content Engines that are running the ACNS 5.4.x software.

<table>
<thead>
<tr>
<th>Table 1-4</th>
<th>Web Clients Supported by Standalone Content Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client Protocol</strong></td>
<td><strong>Client</strong></td>
</tr>
<tr>
<td>HTTP protocol</td>
<td>All Internet browsers including Microsoft Internet Explorer, Netscape, and all others that conform to HTTP 1.0 or HTTP 1.1 specifications.</td>
</tr>
<tr>
<td></td>
<td>Windows Media players, Version 6.x and 7.x, can use MMS-over-HTTP to request content from a Content Engine.</td>
</tr>
<tr>
<td>HTTP</td>
<td></td>
</tr>
<tr>
<td>HTTPS-over-HTTP</td>
<td></td>
</tr>
<tr>
<td>MMS-over-HTTP</td>
<td></td>
</tr>
<tr>
<td>FTP-over-HTTP</td>
<td>Client browsers issuing FTP-over-HTTP requests (support was added in the ACNS 5.1 software release).</td>
</tr>
<tr>
<td>Native FTP</td>
<td>FTP clients (for example, Reflection X clients, WS-FTP clients, or Unix or DOS command-line FTP programs) issuing FTP native requests to the Content Engine that is acting as a nontransparent proxy server for these clients’ FTP native requests (support was added in the ACNS 5.3.1 software release).</td>
</tr>
<tr>
<td>Trivial File Transfer Protocol (TFTP)</td>
<td>TFTP clients (support was added in the ACNS 5.1 software release).</td>
</tr>
</tbody>
</table>
Chapter 1    Introduction

Overview of Standalone Content Engine Capabilities

Table 1-4    Web Clients Supported by Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Client Protocol</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealNetworks proprietary RTSP protocol</td>
<td>RealMedia players use this protocol (the standard IETF RTSP protocol plus proprietary RealNetworks extensions) to request content from the Content Engine. The following media players are collectively referred to as RealMedia players:</td>
</tr>
<tr>
<td></td>
<td>• RealPlayer (Version 8.x and later) from RealNetworks, Inc.</td>
</tr>
<tr>
<td></td>
<td>• RealOne player</td>
</tr>
<tr>
<td>Microsoft proprietary RTSP protocol</td>
<td>Windows Media 9 players use this protocol (the standard IETF RTSP protocol plus proprietary Microsoft extensions) in TCP mode to request content from the Content Engine (supported was added in the ACNS 5.3.1 software release).</td>
</tr>
</tbody>
</table>

Standalone Content Engines can deliver streaming media as live content or as on-demand content (for example, VOD files) to RealMedia players and WMT players. Standalone Content Engines do not support requests from QuickTime players or Cisco IP/TV Viewer.

Supported Caching and Streaming Services with Direct Proxy Routing

Table 1-5 lists the supported caching and streaming services when direct proxy routing is used to direct client request to a standalone Content Engine. An asterisk (*) indicates that the particular caching service can be configured through the Setup utility as well as through the Content Engine CLI.

Table 1-5    Caching and Streaming Services with Direct Proxy Routing

<table>
<thead>
<tr>
<th>Services</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Caching</td>
<td></td>
</tr>
<tr>
<td>HTTP forward proxy caching*</td>
<td>Configuring Nontransparent HTTP Forward Proxy Caching on Standalone Content Engines</td>
</tr>
<tr>
<td>FTP-over-HTTP caching</td>
<td>Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines</td>
</tr>
<tr>
<td>FTP native caching</td>
<td>Configuring Nontransparent FTP Native Caching</td>
</tr>
<tr>
<td>HTTPS proxy caching</td>
<td>Configuring HTTPS Caching for Standalone Content Engines</td>
</tr>
<tr>
<td>WMT Caching and Streaming</td>
<td></td>
</tr>
<tr>
<td>WMT proxy caching*</td>
<td>Enabling and Configuring WMT Caching on Standalone Content Engines</td>
</tr>
<tr>
<td>Streaming of live WMT streaming</td>
<td>Configuring Standalone Content Engines to Deliver WMT Live Streams</td>
</tr>
<tr>
<td>Streaming of preloaded VOD files</td>
<td>Configuring Standalone Content Engines to Distribute VOD Files</td>
</tr>
<tr>
<td>RTSP Caching and Streaming</td>
<td></td>
</tr>
<tr>
<td>RealMedia proxy caching*</td>
<td>Configuring Direct Proxy Routing and RealMedia Proxy Caching</td>
</tr>
<tr>
<td>RealProxy streaming of cached VOD files and live splitting</td>
<td>Configuring RealMedia Services</td>
</tr>
</tbody>
</table>
Supported Caching and Streaming Services with Transparent Redirection

Table 1-6 lists the caching and streaming services that are supported when the standalone Content Engine receives the request through transparent redirection. An asterisk (*) indicates that the particular caching service can be configured through the Setup utility as well as through the Content Engine CLI.

Table 1-6  Caching and Streaming Services with Transparent Redirection

<table>
<thead>
<tr>
<th>Services</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional Caching</td>
<td></td>
</tr>
<tr>
<td>HTTP reverse proxy caching*</td>
<td>Configuring HTTP Reverse Proxy Caching for Standalone Content Engines</td>
</tr>
<tr>
<td>FTP native caching</td>
<td>Configuring Transparent FTP Native Caching</td>
</tr>
<tr>
<td>HTTPS transparent caching</td>
<td>Configuring HTTPS Transparent Caching for Standalone Content Engines</td>
</tr>
<tr>
<td>HTTP transparent caching*</td>
<td>Configuring Transparent HTTP Forward Proxy Caching for Standalone Content Engines</td>
</tr>
<tr>
<td>DNS caching</td>
<td>Configuring the DNS Caching Service (Service 53) for Standalone Content Engines</td>
</tr>
<tr>
<td>WMT Caching and Streaming</td>
<td></td>
</tr>
<tr>
<td>WMT transparent caching*</td>
<td>Enabling and Configuring WMT Transparent Caching on Standalone Content Engines</td>
</tr>
<tr>
<td>Streaming of WMT live streams</td>
<td>Configuring Standalone Content Engines to Deliver WMT Live Streams</td>
</tr>
<tr>
<td>Streaming of preloaded VOD files</td>
<td>Configuring Standalone Content Engines to Distribute VOD Files</td>
</tr>
<tr>
<td>RTSP Caching and Streaming</td>
<td></td>
</tr>
<tr>
<td>RealMedia transparent caching*</td>
<td>Configuring RTSP Transparent Redirection and Caching of RealMedia Requests</td>
</tr>
<tr>
<td>RealProxy streaming of cached VOD files and live splitting</td>
<td>Configuring RealMedia Services</td>
</tr>
</tbody>
</table>

Note

RealProxy is enabled through the Content Engine CLI or through the Setup utility. RealProxy is enabled with a default configuration file. To change the RealProxy default configuration, you must use the RealNetworks RealSystem Administrator GUI. You can use the Content Engine CLI to restore the RealProxy default configuration file on a standalone Content Engine by entering the `rtsp real-proxy default-configuration` EXEC command. For more information, see the “Configuring RealMedia Services” section on page 8-9.
Filtering and Access Control with Standalone Content Engines

Other core capability of standalone Content Engines is the ability to filter and control access to web content. Content Engines can be configured to use their local database or a remote authentication, authorization, and accounting (AAA) server to authenticate and authorize client requests. With the ACNS software (the ACNS 5.2.1 software and later releases), AAA accounting through TACACS+ is also available. (See Chapter 18, “Configuring AAA Accounting on Standalone Content Engines.”)

You can use URL filtering and the Rules Template to block access to any URL or modify the actual content stream (for example, rewrite certain headers). Through the use of access control lists (ACLs), filters can be applied to specific addresses, groups of addresses, or groups of users. In addition to these policies, there are bandwidth limitations and resource controls that determine whether a client request will be accepted at all.

Note

The access control and filtering services that are supported on a standalone Content Engine vary depending on the content protocol (for example, access control is supported for HTTP, HTTPS, and FTP-over-HTTP requests, but ICAP is only supported for HTTP and FTP-over-HTTP). See Table B-5 for a list of the access control and filtering content services that are supported with standalone Content Engines that are running the ACNS 5.4.x software.

In the ACNS 5.2.3 software and later releases, you can also configure a Content Engine to bypass URL filtering for certain HTTP and HTTPS requests. For more information on this feature, see the “Configuring Content Engines to Bypass URL Filtering for Specific HTTP and HTTPS Requests” section on page 11-41.

After the standalone Content Engine receives a client request for content, it performs the following tasks:

- Authenticates the web client, asks the client to provide a username and password so that it can consult an AAA server, and checks whether the client is allowed to access the web. This type of authentication and authorization is referred to as content authentication. For more information, see the “Authentication, Authorization, and Accounting with Standalone Content Engines” section on page 1-16.

- Passes the request through a filter such as Websense or SmartFilter to make sure that the requested object is not objectionable content. For more information, see Chapter 11, “Configuring Content Preloading and URL Filtering on Standalone Content Engines.”

Note

The ACNS 5.x software relies on third-party software to implement content filtering. Supported filtering software includes Websense, N2H2, and SmartFilter.

- Compares content against configured rules, which might rewrite certain headers, redirect the request, or otherwise manipulate the request. See Table 13-2 for a list of the supported rule actions per protocol. For information about how to configure rules on a standalone Content Engine, see Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”

- Checks to see whether the requested content is already in its cache. If so, then the Content Engine serves the object directly from its local cache, rather than from the origin web server, which saves bandwidth to the Internet.

- If the requested content is not already in its cache, the Content Engine retrieves it from the Internet on the client’s behalf, and caches the content for future use if appropriate. The ACNS 5.x software supports this functionality by supporting the web access protocols (including HTTP) and all streaming protocols listed in Table B-1.
Overview of Standalone Content Engine Capabilities

Standalone Content Engines, which are running the ACNS 5.1 software and later releases, also support IP packet filtering, which controls access to specific interfaces (services) on the Content Engine. You can configure IP ACLs that determine whether or not IP packets are allowed to cross specific interfaces on a Content Engine. For example, you can use IP ACLs to control access to content serving and management services on the Content Engine. For more information, see Chapter 19, “Creating and Managing IP Access Control Lists for Standalone Content Engines.”

Authentication, Authorization, and Accounting with Standalone Content Engines

The ACNS 5.x software provides authentication, authorization, and accounting (AAA) support for users who have external access servers (for example, RADIUS or TACACS+ servers), and for users who need a local access database with AAA features.

- **Authentication** (or login) is the action of determining who the user is. It checks the username and password.
- **Authorization** (or configuration) is the action of determining what a user is allowed to do. It permits or denies privileges for authenticated users in the network. For example, if you log in to a standalone Content Engine with a superuser administrator account (for example, the predefined admin account), you have the highest level of access privileges and can perform any administrative task such as the following:
  - Configure the standalone Content Engine.
  - Obtain statistical information that the standalone Content Engine has collected.
  - Reload the device.

  **Note** Generally, authentication precedes authorization and is not mandatory.

- **Accounting** is the action of keeping track of administrative user activities for system accounting purposes. In the ACNS 5.2.1 software and later releases, AAA accounting through TACACS+ is supported. For more information, see Chapter 18, “Configuring AAA Accounting on Standalone Content Engines.”

In ACNS 5.x environments, there are two main types of authentication and authorization:

- **Content authentication**—Controls end user access to content that is served by Content Engines. For more information on this topic, see Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.”

- **Administrative login authentication**—Controls administrative login authentication methods (local, RADIUS, or TACACS+) to process administrator requests to log on to the Content Engine for monitoring, configuration, or troubleshooting purposes.

  An administrator can log in to a standalone Content Engine through the console or the Content Engine GUI. To process these administrative login requests, the Content Engine checks the specified authentication database to verify the user’s username and password and to determine the access rights that this particular administrator should be granted during this login session. When the Content Engine receives a login request, the Content Engine can check its local database or a remote third-party database (the TACACS+ database or the RADIUS database) to verify the username and password and to determine the access privileges of the administrator.
You can configure any combination of these authentication and authorization methods to control administrative login access to a standalone Content Engine:

- Local authentication and authorization
- RADIUS
- TACACS+

By default, the Content Engine uses the local login authentication method as the primary method to process administrative login requests. When you enable local authentication with one or more other authentication methods, local authentication is always attempted first if the priority flags are not set. You cannot specify different administrative login authentication methods for console and Telnet connections. For more information, see Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines.”

**Note**

Content authentication and authorization is independent of administrative login authentication and authorization.

See Table B-4 for a list of the caching, filtering, and authentication mechanisms supported by standalone Content Engines that are running the ACNS 5.4.x software.

### Monitoring and Troubleshooting Features with Standalone Content Engines

It is important that you monitor your Content Engines in order to gauge their performance and to identify any signs that you need to tune their configurations or deploy additional Content Engines. Several tools are available to monitor the performance of standalone Content Engines that are running the ACNS 5.4.x software. This set of tools includes the Cisco Discovery Protocol (CDP), the Simple Network Management Protocol (SNMP), and the ACNS software alarms. For more information on this topic, see the “Monitoring Standalone Content Engines” section on page 21-2.

In addition to monitoring the performance of a Content Engine, transaction monitoring is supported.

**Note**

The term transaction refers to a completed successful or failed request for a web resource by a client. Standalone Content Engines that are running the ACNS 5.x software can record all errors and access activities for reporting purposes.

In the ACNS 5.x software, each content service module (for example, the HTTP module, the WMT server, the FTP proxy process, and the TFTP server) on the Content Engine provides logs of the requests that were serviced. Logging for the following types of requests is provided: HTTP requests, HTTPS requests, FTP requests, WMT requests, RTSP streaming requests, and TFTP requests.

For each content transport protocol, there is a corresponding `show protocol-name statistics` EXEC command that displays the statistics for that protocol.

Typically, Content Engine administrators are interested in what types of requests have been made of the Content Engine and what the results of these requests were. For example, if streaming media is a source of revenue for a company, then the company needs a way to track which customer is accessing which content, how long a user viewed the content, and at what viewing quality. Because these companies charge their customers to stream on-demand content and live broadcasts, they must rely on logged information as the basis for billing their customers for their content access services.
The software logs that record requests that are serviced by a Content Engine are called transaction logs. Typical fields in the transaction log are the date and time when a client 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 address.

Transaction logs are generally used for the following purposes:

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

In the ACNS 5.2.1 software and later releases, Windows Media Services 9 logging is supported. The Windows Media Services 9 Series provides a more robust logging model than Windows Media Services Version 4.1.

You can log data in a predefined format (for example, Squid, Extended Squid, or Apache) or a custom transaction log format that allows you to log additional fields. The contents of the transaction logs can be periodically exported to an external server using FTP. You can also configure log rotation policies.

**Note**

Only one logging format type can be active at a time. When transaction logging is enabled through the Content Engine GUI, the Squid log format is used.

The ACNS 5.x software also supports sanitized logging. If the sanitized logging feature is enabled, the logging of web resource requests do not include (or obfuscate) the identity of the web client. The IP address and usernames of clients in the transaction log file are disguised. For more information, see the “Sanitizing Transaction Logs” section on page 21-42.

In the ACNS 5.2.1 software and later releases, the ability to send HTTP transaction log messages to a remote syslog server is supported. This feature allows you to monitor the remote syslog server for HTTP request authentication failures in real time. For more information, see the “Monitoring HTTP Request Authentication Failures in Real Time” section on page 21-48. For more information about the ACNS 5.x software transaction logs, see the “Monitoring Transactions with Standalone Content Engines” section on page 21-27.

In the ACNS 5.2.3 software and later releases, you can configure a Content Engine to monitor the performance of specific URLs. The Content Engine maintains statistics about the various response characteristics for each of the monitored URLs. For more information on this topic, see the “Monitoring Critical Disk Drives on Standalone Content Engines” section on page 21-17.
Understanding the Basics

This chapter provides an overview of some fundamentals that are important to understand before you configure standalone Content Engines.

This chapter contains the following sections:

- Key Terminology, page 2-2
- Overview of WCCP, page 2-6
- Understanding Some Basic ACNS Software Caching Concepts, page 2-8
- Understanding Some Basic ACNS Streaming Media Concepts, page 2-10
Table 2-1 describes some key terminology that is commonly encountered in caching and streaming.

### Table 2-1  Key Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content-related terms</strong></td>
<td></td>
</tr>
<tr>
<td>Content distribution</td>
<td>Method used to serve content in an ACNS 5.x network. With standalone Content Engines, two distribution methods are supported: pull distribution and preloading. See definitions of preloading and pull distribution in this table.</td>
</tr>
<tr>
<td>Content freshness</td>
<td>Freshness of cached content; that is, whether the cached content has been updated on the origin server since it was cached on the Content Engine. When a user requests on-demand content, the Content Engine checks if a cached object is fresh (if the content has been updated on the origin server since it cached the content locally) before serving the content to the client. If the content has not been modified since the Content Engine has cached the content, the Content Engine sends the client the cached content. If the content has been modified since it was cached, the Content Engine retrieves a fresh copy of the content from the origin server, caches the fresh content, and sends the fresh, cached content to the client. For information about configuring cache freshness, see the “Configuring HTTP Cache Freshness Settings” section on page 7-9.</td>
</tr>
<tr>
<td>Preloading</td>
<td>Content distribution method in which standalone Content Engines retrieve and store specific content in anticipation of future requests for that content. See the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2. Content Engines that are registered with a Content Distribution Manager support pre-positioning as a content distribution method. Standalone Content Engines do not support pre-positioning. For information about pre-positioning content, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.</td>
</tr>
<tr>
<td>Pull distribution</td>
<td>Content distribution method in which content is retrieved and cached on a Content Engine because of a user request (client-triggered demand). Subsequent requests for the same content are served from the local cache. This type of caching is referred to as demand-pull caching.</td>
</tr>
<tr>
<td>Request</td>
<td>Communication from a device (client) to retrieve content.</td>
</tr>
<tr>
<td>Web object</td>
<td>Object that can be transferred over the web (for example, web pages). Collective term used to refer to text objects and binary objects. A text object is an HTML page. A binary object is a web object other than a text object (for example, GIFs and JPEGs).</td>
</tr>
</tbody>
</table>

**Caching-related terms**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caching</td>
<td>Ability to store web objects for later retrieval.</td>
</tr>
<tr>
<td>Cache hit</td>
<td>Requested content (web object) that is in the Content Engine cache.</td>
</tr>
<tr>
<td>Cache miss</td>
<td>Requested content (web object) that is not in the Content Engine cache.</td>
</tr>
<tr>
<td>Live content</td>
<td>Content stream (typically streaming media) that is being broadcast from an origin server. For a more detailed definition, see Table 1-2.</td>
</tr>
</tbody>
</table>
### Key Terminology

#### On-demand content
Content that is acquired, cached, and delivered because of a user request (client-triggered demand). On-demand content is acquired through pull distribution and cached locally on the Content Engine. Pull distribution method is used with the various caching services (for instance, reverse proxy caching, HTTP proxy caching, HTTP transparent proxy caching, RealMedia proxy caching, and RealMedia transparent caching). In contrast, content can be cached on a Content Engine as a result of a preload operation. For a more detailed definition of on-demand content, see Table 1-2.

#### Preloaded content
Content that is retrieved and stored on a standalone Content Engine because you scheduled a retrieval of specific content (a preload operation) in anticipation of user requests for that content. For more information, see the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2.

### Device-related terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>End user who is using a browser or media player to request web content. Also called web clients. See the “Web Clients Supported by Standalone Content Engines” section on page 1-12.</td>
</tr>
<tr>
<td>Origin server</td>
<td>Server that contains the content that the client requested. This server acts as the master content repository. In the case of streaming media, the origin server receives live streaming from an encoder. These streams are then forwarded or split to Content Engines that act as edge devices that receive content from an origin streaming server. In the case of nonstreaming content, the Content Engine retrieves the requested content from the origin server if the content is not already stored in the local cache.</td>
</tr>
</tbody>
</table>
| Standalone Content Engines    | Content Engines that are running the ACNS 5.x software and are intentionally not registered with a Content Distribution Manager (if there is one) so that they can be managed as a standalone devices. Standalone Content Engines can be configured or managed using the following methods:  
  - The Setup utility  
  - The Content Engine CLI  
  - The Content Engine GUI  
  Multiple standalone Content Engines can be deployed (for example, you can deploy clusters of standalone Content Engines.)  
  This guide describes how to configure, manage, and monitor standalone Content Engines as caching and streaming engines.  
  For information about Content Engines that are registered with a Content Distribution Manager, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.* |

### Deployment-related terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
</table>
| Centrally managed deployments| Deployments that consist of one or more of the following ACNS 5.x network devices: Content Distribution Managers, Content Engines that are registered with a Content Distribution Manager, and Content Routers that are running the ACNS 5.x software.  
  For information about configuring, maintaining, and monitoring such deployments, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.* |
### Table 2-1  Key Terminology (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locally managed deployments</td>
<td>Deployments that consist of one or more standalone Content Engines that are running the ACNS 5.x software and are configured as caching and streaming engines. These types of deployments are the focus of this guide.</td>
</tr>
<tr>
<td>PAC file</td>
<td>File (proxy automatic configuration) that is used to point client browsers to a specific Content Engine as their proxy server. This file resides on a server in your intranet, and can be loaded at startup by a browser on a client desktop. A PAC file is written in JavaScript. Use a PAC file if you want to use direct proxy routing to direct request from certain clients to a specific Content Engine. For more information, see the “Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine” section on page 4-37.</td>
</tr>
</tbody>
</table>

### Routing-related terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct proxy routing</td>
<td>Type of routing in which client browsers or media players are configured to explicitly point to a specific forward proxy server (the Content Engine). Because the web clients are aware that their content requests are being directed to the Content Engine (forward proxy server), the clients can point to this Content Engine for nontransparent caching services (for example, client acceleration by quickly accessing cached content on this Content Engine). See Table 1-5 for a list of supported caching and streaming services available when direct proxy routing is used. For information about configuring clients to use this routing method, see the “Configuring Client Browsers and Media Players for Direct Proxy Routing” section on page 4-35.</td>
</tr>
<tr>
<td>Transparent redirection</td>
<td>Type of routing in which Web Cache Communication Protocol (WCCP) router or Layer 4 switch transparently intercepts client requests and redirects the request to the Content Engine. The web clients are not aware that a WCCP router or a Layer 4 switch is redirecting their requests to the Content Engine (the transparent proxy server). See Table 1-6 for a list of supported caching and streaming services when transparent redirection is used. For information about configuring this routing method, see Chapter 6, “Configuring Transparent Redirection for Standalone Content Engines.”</td>
</tr>
</tbody>
</table>
### Key Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proxy server-related terms</strong></td>
<td></td>
</tr>
<tr>
<td>Forward proxy server</td>
<td>Content Engine that receives content requests directly from the web clients because direct proxy routing is being used to direct client requests to the Content Engine. See the “Overview of Nontransparent Forward Proxy Caching” section on page 3-3.</td>
</tr>
<tr>
<td>Reverse proxy server</td>
<td>Content Engine that is running the reverse proxy service for web server acceleration. Acts as a proxy for a small number of web servers (web server farms) by locally caching the requested content from these specific web servers. See the “Overview of Transparent Reverse Proxy Caching” section on page 3-10.</td>
</tr>
<tr>
<td>Transparent proxy server</td>
<td>Content Engine that is used for client acceleration. Acts as a proxy for web clients by caching requested content whenever possible in its local cache. Used if transparent redirection (WCCP routing or Layer 4 switching) is the routing method used to direct client requests to the Content Engine. The web clients are not aware that their requests are being intercepted by a WCCP router or a Layer 4 switch and being redirected to the Content Engine (the transparent proxy server). See the “Overview of Transparent Forward Proxy Caching” section on page 3-6.</td>
</tr>
<tr>
<td><strong>Service-related terms</strong></td>
<td></td>
</tr>
<tr>
<td>Proxy cache services</td>
<td>Caching services that use a Content Engine as a proxy server for the following types of acceleration:</td>
</tr>
<tr>
<td></td>
<td>• Client acceleration for HTTP, HTTPS, FTP, WMT, and RTSP requests (through a Content Engine that is functioning as a forward proxy server or transparent proxy server)</td>
</tr>
<tr>
<td></td>
<td>• Web server acceleration for HTTP or FTP-over-HTTP requests (through a Content Engine that is functioning as a reverse proxy server)</td>
</tr>
<tr>
<td>RTSP streaming and caching services</td>
<td>Supported RTSP streaming and caching services (for example, RealMedia proxy caching, RealMedia transparent caching, and RealProxy live splitting). For more information, see Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”</td>
</tr>
<tr>
<td>Sanitized logging</td>
<td>Logging of web resource requests that does not include (or deliberately hides) the identity of the client. For more information, see the “Sanitizing Transaction Logs” section on page 21-42.</td>
</tr>
<tr>
<td>Transaction</td>
<td>Completed request (whether successful or failed) for a web resource by a client. Information about transactions is recorded in the ACNS software transaction logs. For more information, see the “Monitoring Transactions with Standalone Content Engines” section on page 21-27.</td>
</tr>
<tr>
<td>WCCP</td>
<td>Cisco Web Cache Communication Protocol (WCCP). Protocol used by routers and Content Engines to support transparent redirection of client requests for content to Content Engines that are functioning as transparent proxy servers. For more information, see the “Overview of WCCP” section on page 2-6.</td>
</tr>
<tr>
<td>WMT streaming and caching services</td>
<td>Supported WMT streaming and caching services (for example, WMT proxy caching, WMT transparent caching, and WMT live splitting). For more information, see Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”</td>
</tr>
<tr>
<td><strong>Streaming-related terms</strong></td>
<td></td>
</tr>
<tr>
<td>RealMedia (RM) format</td>
<td>Format of RealNetworks streaming media files. Standalone Content Engines can serve RealMedia content (.rm files) to RealMedia players if RealProxy is enabled on the Content Engine. For more information, see Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”</td>
</tr>
</tbody>
</table>
Overview of WCCP

Cisco developed the Web Cache Communication Protocol (WCCP) within Cisco IOS software to enable routers or switches to transparently redirect packets to network caches. In environments with standalone Content Engines, WCCP acts as a communication mechanism between a router and the standalone Content Engines.

WCCP is only licensed on the Content Engine (caching device) and not on the redirecting router. WCCP does not interfere with normal router or switch operations. The router must be running a version of Cisco IOS software that supports WCCP Version 1 or Version 2.

When caching support is enabled on the router and WCCP support is enabled on the Content Engines, the devices can communicate and deliver the WCCP features and services for which they are configured. 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.

Two versions of WCCP are available: Version 1 and Version 2.

Table 2-1  Key Terminology (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMIL</td>
<td>Synchronized Multimedia Integration Language. An open standard markup language that is similar to HTML. This simple but powerful markup language allows you to coordinate multiple clips. SMIL also allows you to define how, when, and where you want the multiple clips to be played. The SMIL file establishes the overall time line of the presentation.</td>
</tr>
<tr>
<td>RealMedia players</td>
<td>RealNetworks media players such as RealPlayer and RealOne player. If a standalone Content Engine receives a request from a RealMedia player (either through direct proxy routing or transparent interception), the Content Engine uses the RealNetworks RTSP protocol to serve the requested RealMedia content to the RealMedia player. If the requested RealMedia content is not already stored in its local cache, the Content Engine retrieves the requested content from the origin streaming server (RealNetworks server). For more information, see Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”</td>
</tr>
<tr>
<td>Stream splitting</td>
<td>Process in which a single live stream from the origin server is split, or copied, into multiple streams by a splitter to serve to clients that requested the stream. This process is especially common when the stream has to traverse a mixture of multicast and nonmulticast-enabled networks. The splitter can receive incoming streams by unicast or multicast, and can serve streams by unicast or multicast. Stream splitting can be used for live streams or scheduled rebroadcasts. For information about stream splitting with WMT, see the “Configuring Standalone Content Engines to Deliver WMT Live Streams” section on page 9-37.</td>
</tr>
</tbody>
</table>
The following is a summary of the capabilities and limitations of WCCP Version 1:

- Only a single WCCP-enabled router (home router) is supported.
- Only a single WCCP service (the web-cache service [service 0]) is supported.
- Traffic redirection is only supported on port 80.
- Each home router can support the standard web-cache service (service 0) for up to 32 Content Engines.
- There is no bypass support (for example, static bypass, error bypass, and authentication bypass are not supported).
- There is no support for generic routing encapsulation (GRE) on return.

WCCP Version 2 supports a broad set of services and features (including bypass support) and more than one WCCP-enabled router. (See Table 6-3 for a list of WCCP Version 2 features and services.) Consequently, we recommend that you use WCCP Version 2.

The following sequence of events details the interaction between Content Engines and routers that have been configured to run WCCP Version 2:

1. Each Content Engine is configured with a router list. (See the “Defining Router Lists on Standalone Content Engines” section on page 6-12.)

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.

   Routers and Content Engines become aware of one another and form a WCCP service group using a management protocol. The Content Engines also send periodic “Here I am” messages to the routers that allow the routers to rediscover the Content Engines. To properly depict the view, the protocol needs to include the list of routers in the service group as part of its messages.

3. Once the view is consistent across all the Content Engines in the Content Engine cluster, one Content Engine is designated the lead. When there is a cluster 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 lead Content Engine is to determine how traffic should be allocated across the Content Engines in the Content Engine cluster. The lead Content Engine sets the policy that the WCCP-enabled routers must adhere to when redirecting packets to the Content Engines in this cluster. The assignment information is passed to the entire service group from the designated Content Engine so that the routers in the service group can redirect the packets properly and the Content Engines in the service group can better manage their load.

When you use the Setup utility to configure HTTP, WMT, or RealMedia transparent caching through WCCP redirection, WCCP Version 2 is used. In order to use WCCP Version 1 for HTTP transparent caching (the web-cache service), you must configure this WCCP service through the Content Engine CLI. (See the “Example 1—Configuring the Web-Cache Service with WCCP Version 1” section on page 6-39.) For information about configuring transparent redirection with WCCP, see Chapter 6, “Configuring Transparent Redirection for Standalone Content Engines.”
Overview of WCCP Services

Some of the WCCP services that routers can supply have a well known set of criteria and a predefined service identifier (for example, the web-cache service has zero (0) as its identifier [service 0]). Other examples of such services include the reverse-proxy service (service 99), the HTTPS-cache service (service 70), and the RTSP service (service 80).

Other services that are not well known can be supported by defining a dynamic (user-defined) WCCP service (services 90 to 97) using WCCP Version 2. As part of defining a dynamic WCCP service, you specify a set of criteria (for example, the application type [HTTP caching, HTTPS caching, or streaming], hash parameters, password, and weight), and assign this service a service identifier (service 90 to 97) on the Content Engine. Although you can specify the criteria for a user-defined service on the Content Engine, the WCCP Version 2-enabled routers must support the particular service identifier (service 90 to 97). For more information about configuring user-defined WCCP Version 2 services, see the “Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 6-15.

WCCP Version 1 only supports one WCCP service (the web-cache service) and a single router (home router). Only WCCP Version 2 supports user-defined WCCP services (services 90 to 97). Consequently, we recommend that you use WCCP Version 2 because it supports a wider set of features and services (see Table 6-3) as well as multiple routers.

See Table B-3 for a list of supported WCCP services. For information about using the Content Engine CLI to configure WCCP services for standalone Content Engines, see Chapter 6, “Configuring Transparent Redirection for Standalone Content Engines.”

Understanding Some Basic ACNS Software Caching Concepts

This section describes the following fundamentals about ACNS software caching:

- Introduction to Caching, page 2-8
- Cacheable Content, page 2-10
- Network Protocols and Caching, page 2-10

Introduction to Caching

Caching refers to the ability to store web objects such as web pages for later retrieval. Typically a web client requests an object from a web server using a web browser. (See Figure 2-1.)

![Figure 2-1 Basic Web Request](image)

Every web object has a uniquely defined address (URL) that allows the web browser to retrieve web objects, assuming that the server address exists in the DNS space. The basic components of a URL address are shown in Figure 2-2.
To implement caching using web browsers, browsers have a local cache that the user can set up to help retain recently viewed pages for easier access during the web-viewing process. (See Figure 2-3.) For instance, clicking the Back button on a browser takes advantage of local caching because this recently viewed page is rapidly displayed.

Despite the inherent advantages obtained using a local cache, a Content Engine is often used to provide specialized caching features to many users.

The ability of the Content Engine to store information on a large scale for later retrieval has significant advantages for the user and Internet traffic as a whole:

- It reduces network congestion by keeping web objects close to the user instead of accessing the same content through the network.
- It reduces the time it takes to display a web page, because the page is stored locally or close to the user.
- It reduces the load on the origin web server, because this server does not have to redistribute content that has recently been acquired.
In this scenario, a user attempts to retrieve a web object from a web server. Because the local browser
does not have the page stored in its cache, the browser sends the request to the origin web server using
the Content Engine as its proxy for the web request. In this deployment, the Content Engine serves as a
proxy to the web client, and it first tries to satisfy the web request from its cache of stored objects.

**Cacheable Content**

Cacheable content is typically static application data and can be associated with a file type and file
extension, as described in Table 2-2.

<table>
<thead>
<tr>
<th>File Type</th>
<th>File Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphic files</td>
<td>gif, jpeg, bmp</td>
</tr>
<tr>
<td>Compressed files</td>
<td>zip, gz, tar</td>
</tr>
<tr>
<td>Document files</td>
<td>text, txt, pdf</td>
</tr>
<tr>
<td>Multimedia files</td>
<td>avi, mov, mpeg, wav</td>
</tr>
</tbody>
</table>

**Network Protocols and Caching**

The interaction between a web browser and a web server uses the existing standard application-layer
Internet protocols such as HTTP and RTSP. The Content Engine has to be able to serve web objects to
the web client using all of these web access protocols. See Table B-1 for a descriptive list of the network
protocols that a Content Engine, which is running the ACNS 5.4.1 software and later releases, can use
to serve content to web clients.

**Understanding Some Basic ACNS Streaming Media Concepts**

This section describes how streaming media services are supported on a Content Engine in a locally
managed deployment, and discusses the following topics:

- Understanding Some Basic ACNS Streaming Media Concepts, page 2-10
- Understanding IP Multicasting Fundamentals, page 2-16

For a list of supported streaming media protocols, see Table B-2. For information about how to configure
media services on standalone Content Engines, see:

- Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines”
- Chapter 8, “Configuring RealMedia Services on Standalone Content Engines”

For information about how to configure media services on Content Engines that are registered with a
Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally
Managed Deployments, Release 5.5.
Understanding Streaming Media Solutions

Streaming is a technology that allows content to be accessed or viewed before all the media packets have been received, with caching, the content must be received in its entirety before it can be accessed. Streaming media can be delivered as live content or on-demand content, such as video on demand (VOD).

Cisco ACNS software supports several types of streaming media solutions, including the Windows Media Technologies (WMT) solution from Microsoft Corporation and the RealMedia solution from RealNetworks, Inc. In the ACNS 5.3.1 software release, support for the Windows Media 9 RTSP server was added. See Table 1-3 for a list of supported streaming media solutions.

The type of streaming media solutions that you can deploy varies based on whether the Content Engine is registered with a Content Distribution Manager or is a standalone Content Engine. For example, the RTSP-based Cisco Streaming Engine and Apple QuickTime solutions are not supported on standalone Content Engines.

RTP and RTSP are two Internet standard protocols that are designed for streaming media over the Internet.

- RTP (RFC-1889) is the data transfer protocol for multimedia data over both unicast (TCP and UDP) and multicast.
- RTSP is a standard Internet streaming control protocol (RFC-2326).

RTSP is an application-level protocol that controls the delivery of streaming media data with real-time properties, such as video and audio. RTSP is a text-based protocol, and typically runs over TCP port 554. RTSP acts as a network remote control for multimedia servers and has been widely adopted as a streaming media protocol. Apple QuickTime, RealProxy, and Cisco Streaming Engine are all content distribution methods that use RTSP as the streaming media protocol, and RTP as the data transfer protocol. Real Networks added proprietary extensions to the standard RTSP protocol (for example, the challenge and response between a RealMedia client and a RealNetworks streaming server).

Support for the IETF standard RTP and RTSP protocols is included as part of the ACNS 5.1 software and later releases. However, separate licenses are required for the following two streaming media features that can be enabled on a Content Engine that is running the ACNS 5.x software:

- The RealProxy streaming media feature requires a RealProxy license. For more information on this topic, see the “Configuring RealMedia Services” section on page 8-9.
- The WMT streaming media feature requires a WMT license. For more information about the WMT streaming solution for standalone Content Engines, see the “Configuring WMT Streaming Media Services on Standalone Content Engines” section on page 9-1.

For information about how to configure streaming media services for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

Figure 2-4 shows streaming media files, which 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. 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.
In Figure 2-4, both audio and video are being recorded from an event to be distributed later either as a 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. The streaming media server in turn delivers the media files on a live or on-demand basis to the Content Engine, which then relays the requested content to users who are running a particular media software (for example, the Windows Media player) on their personal computers or other electronic devices.

Client requests for media files can be directed to the Content Engine through the following routing methods:

- With direct proxy routing, the client media players are configured to point directly to the Content Engine.
- With transparent redirection, a WCCP router or Layer 4 switch transparently intercepts a client’s request for media files and transparently redirects the request to the Content Engine.

In both types of routing, the Content Engine always obtains the live content from the external streaming media server (Encoder > streaming media server > Content Engine) or encoder (Encoder > Content Engine); the Content Engine is not the originator of the live content.

Browsers mainly use HTTP to fetch contents, and streaming media players mainly use various streaming protocols to fetch and display contents. For example, RealMedia players use the RealNetworks RTSP protocol to stream audio and video contents, the Windows Media player uses the MMS-over-HTTP to stream audio and video contents, and QuickTime players use the IETF standard RTSP protocol to stream multimedia contents. The media player on the client desktop can be started directly by the end user, or it can be automatically launched by the client browser. See Table 1-4 for a list of supported web clients.
How Client Browsers Start Media Players on Client Desktops

It is important to understand the ways in which media players are launched, because media-index files and relative links impose restrictions on the content distribution software, typically requiring that the directory structure be maintained on the Content Engine.

The following are the two primary ways for the client browser to spawn a streaming media player on a client desktop:

- Through Java script embedded in the web page.
- For the browser to start a streaming media player through a MIME-type association. An MIME-type association is a browser capability that enables the browser to invoke a particular application when it encounters an object with a particular MIME-type suffix. A set of default association rules covers the common object types on the Internet. Users can edit, add, or delete these MIME-type association rules in their browsers. For example, through MIME-type association, the client browser launches the Adobe Acrobat reader when it encounters a *.pdf file, and it launches the Windows Media Player when it encounters an *.asf or *.asx file.

Client browsers can launch media players on a client desktop in a variety of situations. For example, if the HTML page contains a link to a video that is encoded in ASF format (see example), while processing this HTML page, the client browser launches the Windows Media player on the client desktop in order to retrieve the file wmt_vod/welcome1.asf from the server named stream-ce50.abc.com. MMS-over-HTTP is used to retrieve the .asf file for the Windows Media player.

```html
<html>
<head>
<title>Example Page with a Video</title>
</head>
<body>
<a href="rtsp://stream-ce50.abc.com/wmt_vod/welcome1.asf">TEST CLIP</a>
</body>
</html>
```

A more common example of a how a client browser launches the media player on the client desktop is when the video and presentation material is packaged in a media-index file such as an .asx file or a Synchronized Multimedia Integration Language (SMIL) file. The .asx file is used by WMT streaming, and the SMIL file is used by the RealNetworks RealMedia streaming. The browser is typically configured so that the moment it retrieves an .asx file, it spawns the Windows Media Player and passes the .asx file to the player. Similarly, when the client browser retrieves an SMIL file, it starts the RealMedia player on the client desktop and passes the SMIL file to the RealMedia player on the client desktop.

**Note**

In the ACNS 5.2.1 software release, support for SMIL files for RealProxy was added.
Media-index files can contain either relative links to media files or absolute links to media files. For example, the following is an .asx file that contains absolute links to .asf files:

```xml
<!--A simple playlist with entries to be played in sequence.-->
<ASX version = "3.0">
  <Title>The Show Title</Title>
  <Entry>
    <Ref href = "rtsp://adventure-works.com/Path1/title1.wma" />
  </Entry>
  <Entry>
    <Ref href = "rtsp://abc-studio.com/Path2/title2.wma" />
  </Entry>
  <Entry>
  </Entry>
</ASX>
```

The .asx file instructs the player to retrieve three videos, which are at different websites, in the specified sequence. The full URL for the video is listed in the .asx file.

RealMedia files have an .rm extension, which indicates that the file is in a RealMedia format.

In the case of a relative link, the full URL is constructed by replacing the last path component in the SMIL file’s URL path with the relative link.

Most media players support synchronized media, that is, audio and video that are synchronized with a slide presentation. The media are typically put in the same directory, and are referenced using relative links.

### How a Client Media Player Issues a Request

After the media player has been launched, it is given a media file to play and the player follows procedures similar to those of the browser:

1. The media player looks up the IP address of the streaming server through the DNS resolution server that is configured on the client desktop. If a proxy is configured for the media player, the media player does not perform this DNS lookup; the player requests that the proxy perform the look up for the IP address. Just as you can configure a client browser to point directly to a Content Engine, you can also configure client media players to point directly to a Content Engine. For more information, see Pointing RealMedia Players Directly to a Standalone Content Engine, page 4-46

2. The media player tries to establish a connection to the media server on the default port (port 1755 for WMT, and port 554 for RTSP) or a specified port. In the case of a proxy, it will establish a connection to the proxy on the specified port.

3. Once the connection is established between the media player and the media server or the proxy, the media player exchanges information with the media server or the proxy to establish the sessions.
Typically, the first packet that the media player sends to the server or the proxy contains the IP address of the intended media server or the domain of the server. This is different from HTTP, where the first packet contains not only the domain information but also the object information. In the case of both WMT and RTSP streaming protocols, the first packet carries server information only. The object information (pathname) is not available until the second or the third packet exchange between the client media player and the server. When the WMT protocol is used, the object information is obtained in the fourth packet exchange. When RTSP-over-RTP is used, the object information is obtained in the third packet exchange.

**RTSP Redirect Method**

Similar to HTTP, the RTSP protocol also has a redirect message. The server can send a redirect message to the media player with a new URL, and the player plays the new URL.

**WMT Redirect Method**

The current version of the WMT streaming protocol does not provide a redirect method. Instead, a more commonly used method is for the server to use the .asx file to publish videos, and, if redirection is needed, for the server to rewrite the .asx file (changing the references in the file). As explained previously (“How Client Browsers Start Media Players on Client Desktops”), the .asx file directs the media player to retrieve various content objects, so rewriting the references in the file effectively redirects the media player to different URLs.

**Understanding Caching Policies in Streaming Media Caching**

In contrast to HTTP caching, caching policies in streaming media caching are much simpler, because streaming media are mostly large static content.

With WMT caching, you use the Content Engine GUI or CLI (for example, the `wmt cache max-obj-size` global configuration command) to specify the caching policies for a standalone Content Engine. All responses are cacheable, including partial responses. All WMT requests result in communication between the Content Engine and the origin server, whether the Content Engine establishes the streaming control session with the server for the client, even if the request is a cache hit. By establishing the streaming control session, the Content Engine can verify that its cached content is fresh, and the client can access the content. Because streaming objects are typically very large, the overhead of establishing the control session with the server is negligible and does not reduce the bandwidth savings from the cache hits.

With RealProxy caching (RealMedia proxy caching and RealMedia transparent caching), you specify the RealProxy caching policies through the RealNetworks RealSystem Administrator GUI. For more information on this topic, see the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21.
Understanding IP Multicasting Fundamentals

Multicast delivery enables the distribution of streaming media by allowing different receiving 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.

To take advantage of multicasting, all devices (including Content Engines, routers, and clients) must be multicast-enabled. For this reason, multicasting is mostly used in local networks where routers can be configured for multicasting. Multicast delivery over the Internet can only be accomplished when all the devices that participate in the multicast have been enabled for multicasting.

The Internet Assigned Numbers Authority (IANA) controls the assignment of IP multicast addresses. The IANA has assigned the IPv4 Class D address space to be used for IP multicast. Therefore, all IP multicast group addresses fall in the range from 224.0.0.0 through 239.255.255.255. However, some combinations of source and group address should not be routed for multicasting purposes. See the “Unusable Multicast Address Assignments” section on page B-11 for a list of the unusable multicast address ranges and the reasons they should not be used.

Insecure Services

Applications that use multicast addresses in the 224.0.1.2/32, 224.0.1.22/32, and 224.0.2.2/32 ranges have been demonstrated to be vulnerable to exploitation, which has led to serious security problems.

Copying Files Between Servers and Clients

Some applications are used to copy files between servers and clients and to otherwise maintain groups of personal computers. These applications are intended to be used on a local subnet or within an administrative domain, but the default addresses used by the software are not within the administratively scoped addresses listed in Table B-8 on page B-12.

Limited Scope Addresses

Limited scope addresses are also called administratively scoped addresses. These addresses are described in RFC 2365, *Administratively Scoped IP Multicast*, to be restricted to a local group or organization. Companies, universities, or other organizations can use limited scope addresses to have local multicast applications that will not be forwarded outside their domain. Routers typically are configured with filters to prevent multicast traffic in this address range from flowing outside an autonomous system (AS) or any user-defined domain. Within an autonomous system or domain, the limited scope address range can be further subdivided so that local multicast boundaries can be defined. This subdivision is called address scoping and allows for address reuse between these smaller domains.

Source-Specific Multicast Addresses

Addresses in the 232.0.0.0/8 range are reserved for source-specific multicast (SSM). SSM is an extension of the Protocol-Independent Multicast (PIM) protocol that allows for an efficient data delivery mechanism in one-to-many communications.
GLOP Addresses

RFC 2770, *GLOP Addressing in 233/8*, proposes that the 233.0.0.0/8 address range be reserved for statically defined addresses by organizations that already have an AS number reserved. This practice is called GLOP addressing. The AS number of the domain is embedded into the second and third octets of the 233.0.0.0/8 address range. For example, the AS number 62010 is written in hexadecimal format as F23A. Separating the two octets F2 and 3A results in 242 and 58 in decimal format. These values result in a subnet of 233.242.58.0/24 that would be globally reserved for AS 62010 to use.

Layer 2 Multicast Addresses

Historically, network interface cards (NICs) on a LAN segment could receive only packets destined for their burned-in MAC address or the broadcast MAC address. In IP multicast, several hosts need to be able to receive a single data stream with a common destination MAC address. Some method had to be devised so that multiple hosts could receive the same packet and still be able to differentiate between several multicast groups.

One method is to map IP multicast Class D addresses directly to a MAC address. Today, using this method, NICs can receive packets destined to many different MAC addresses—their own unicast, broadcast, and a range of multicast addresses.

The IEEE LAN specifications made provisions for the transmission of broadcast and multicast packets. In the 802.3 standard, bit 0 of the first octet is used to indicate a broadcast or multicast frame. *Figure 2-5* shows the location of the broadcast or multicast bit in an Ethernet frame.

*Figure 2-5* IEEE 802.3 MAC Address Format

<table>
<thead>
<tr>
<th>Octet 0</th>
<th>Octet 1</th>
<th>Octet 2</th>
<th>Octet 3</th>
<th>Octet 4</th>
<th>Octet 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

This bit indicates that the frame is destined for a group of hosts or all hosts on the network (in the case of the broadcast address 0xFFFF.FFFF.FFFF).

IP multicast makes use of this capability by sending IP packets to a group of hosts on a LAN segment.

Note

The MAC address table on an ACNS 5.x Content Engine supports up to 50,000 entries.

Ethernet MAC Address Mapping

The IANA owns a block of Ethernet MAC addresses that start with 01:00:5E in hexadecimal format. Half of this block is allocated for multicast addresses. The range from 0100.5e00.0000 through 0100.5e7f.ffff is the available range of Ethernet MAC addresses for IP multicast.

This allocation allows for 23 bits in the Ethernet address to correspond to the IP multicast group address. The mapping places the lower 23 bits of the IP multicast group address into these available 23 bits in the Ethernet address. (See *Figure 2-6*.)
Because the upper five bits of the IP multicast address are dropped in this mapping, the resulting address is not unique. In fact, 32 different multicast group IDs map to the same Ethernet address. (See Figure 2-7.) Network administrators should consider this fact when assigning IP multicast addresses. For example, 224.1.1.1 and 225.1.1.1 map to the same multicast MAC address on a Layer 2 switch. If one user subscribed to Group A (as designated by 224.1.1.1) and the other users subscribed to Group B (as designated by 225.1.1.1), they would both receive both A and B streams. This situation limits the effectiveness of this multicast deployment.
Deployment Scenarios for Standalone Content Engines

This chapter introduces some sample scenarios for deploying standalone Content Engines in enterprise and service provider environments. This chapter contains the following sections:

- Deciding Which Services to Deploy, page 3-2
- Deploying Caching and Streaming Services in Nontransparent Mode, page 3-3
- Deploying Caching and Streaming Services in Transparent Mode, page 3-6
- Deploying Streaming Media Services on Standalone Content Engines, page 3-14
- Deploying Filtering and Access Control Services, page 3-15

\[\text{Note}\]

For information about deploying a Content Engine as a device that is registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Deciding Which Services to Deploy

By pushing content to the edges of a network, you can accelerate content delivery and optimize WAN bandwidth usage. The process used to accomplish this is called content caching. Content caching is also referred to as network caching. Because of the special position of the Content Engine as an in-line device between the web clients and the Internet, you can easily deploy Content Engines as content caching engines. Bandwidth usage and web latency are significantly reduced, because frequently accessed Internet content is being locally cached and served by the Content Engine at each location.

**Note**
To integrate with existing proxy infrastructures, the ACNS software supports a number of proxy protocols, including FTP, HTTPS, HTTP 1.0, and HTTP 1.1. See Table B-1 for a list of supported network protocols.

The types of supported services that you can deploy with a Content Engine vary depending on the method used to route the request to the Content Engine. Content requests can be routed to the Content Engine directly from clients (direct proxy routing) or through a WCCP router or Layer 4 CSS switch (transparent redirection).

**Note**
See Table 1-5 and Table 1-6 for a list of caching and streaming services supported when direct proxy routing or transparent redirection is used to direct client requests to a standalone Content Engine.

The direct proxy routing method is the simplest, most straightforward routing method. With direct proxy routing, you must configure client desktops (client browsers and media players) to send their content requests directly to a specific Content Engine that is functioning as their proxy server. Client requests are sent directly to the Content Engine that is configured as the client’s proxy server. This routing method is typically used when user desktops are tightly controlled.

The transparent redirection method requires an understanding of network topology and traffic patterns. Organizations generally prefer to use the transparent redirection method because it does not require any configuration changes to client desktops.

However, even though direct proxy routing requires changes to client desktops, there may be a legacy requirement for using direct proxy routing. There also may be cases in which organizations need to use direct proxy routing for a particular service (for example, HTTPS proxy caching) because they are not allowed to make the necessary configuration changes to the WCCP router or switch at the branch office.

This section introduces the routing methods and the associated caching and streaming services that are supported by standalone Content Engines that are running the ACNS 5.x software:

- Deploying Caching and Streaming Services in Nontransparent Mode, page 3-3
- Deploying Caching and Streaming Services in Transparent Mode, page 3-6

**Note**
A standalone Content Engine supports direct proxy routing and transparent redirection through WCCP routing and Layer 4 switching. However, if you want to use content routing as a routing method, then you must register the Content Engine with a Content Distribution Manager. Standalone Content Engines are not registered with Content Distribution Managers and therefore cannot support Content Routing. For information about Content Routing, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Deploying Caching and Streaming Services in Nontransparent Mode

With direct proxy routing, the standalone Content Engine is the destination of all browser or media player requests for web content. Such requests are called proxy-style requests. A proxy-style request arrives with the same destination IP address as that of the Content Engine; the request is specifically routed directly to the Content Engine (the forward proxy server) by the web client.

Overview of Nontransparent Forward Proxy Caching

In deployments that use direct proxy routing to route content requests to the Content Engine (see Figure 3-1), the Content Engine acts as a network gateway device that is optimized to retrieve content on behalf of web clients.

- If the requested content is already in the Content Engine’s local storage (cache hit), the Content Engine sends the content to the web client.
- If the requested content is not already stored in the Content Engine’s local cache (cache miss), the Content Engine retrieves the requested content from the origin server, stores a local copy of the content if the content is cacheable, and sends the requested content to the web client. When the Content Engine receives subsequent requests for the same content, it serves the content from its local storage.

Figure 3-1 Nontransparent Forward Proxy Caching with Standalone Content Engines

1. Client browsers and media players are configured to request content directly from Content Engine
2. Web clients send content requests directly to Content Engine
3a. If cache hit, then 4
   3b. If cache miss, Content Engine performs DNS request to determine IP address of origin server, retrieves and caches the content, and sends to client (4)
4. Requested content sent to web client
   (nontransparent forward proxy server for web clients)
Typically, direct proxy routing is implemented in enterprise environments as opposed to service provider environments because this type of caching requires modifications to client desktops. With direct proxy routing, you must explicitly configure client browsers and media players to point to the Content Engine that is functioning as a forward (nontransparent) proxy for these clients. For more information on this topic, see the “Configuring Client Browsers and Media Players for Direct Proxy Routing” section on page 4-35.

Some significant advantages to deploying nontransparent (proxy) caching services on standalone Content Engines are:

- Internet access for user populations can be regulated by the Content Engine that is acting as a gateway device for these end users.
- Internet requests all appear to be sourced from the proxy cache (Content Engine), which hides internal network addressing.
- Frequently requested cacheable content is cached locally, which results in significant WAN bandwidth savings and accelerated delivery of content to web clients.

In the ACNS 5.2 software release, the Setup utility was added to expedite the basic configuration (device network settings, disk configuration, and a set of commonly used caching services) of standalone Content Engines. This basic configuration includes a set of commonly used caching services. See Table 4-2 for a list of caching services that you can configure with the Setup utility.

### Configuring Incoming Proxy Ports for Nontransparent Mode Services

In proxy mode, the Content Engine that is functioning as a forward proxy server supports up to eight incoming ports each for FTP, HTTP, HTTPS, and RTSP requests. The term *proxy port* is used to refer to the port from which the Content Engine receives the proxy-style request and sends the requested content back to the client. The incoming proxy ports can be the same ports that are used by transparent mode services (for example, HTTP transparent caching). You can change the incoming proxy ports on the Content Engine without stopping any WCCP services that are running on the Content Engine.

As part of configuring a nontransparent mode service (for example, HTTP proxy caching, FTP-over-HTTP proxy caching, RealMedia proxy caching, and WMT proxy caching), you must do the following:

- Configure the client browser or media player to direct any requests for request to the incoming proxy port on the Content Engine.
- Configure the Content Engine to listen on the incoming proxy port for client requests.

A standalone Content Engine accepts nontransparent (proxy-style) requests directly from a client browser or media player when the incoming proxy ports are configured on the client browser and media player and on the Content Engine.
For information about how to point client browsers or media players to a specific Content Engine, see Table 3-1.

**Table 3-1 Configuring Client Browsers and Media Players to Support Direct Proxy Routing of Content Requests**

<table>
<thead>
<tr>
<th>Nontransparent Caching</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP proxy caching</td>
<td>See the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.</td>
</tr>
<tr>
<td>WMT proxy caching of WMT RTSP requests</td>
<td>See the “Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests” section on page 4-43.</td>
</tr>
<tr>
<td>RealMedia proxy caching</td>
<td>See the “Pointing RealMedia Players Directly to a Standalone Content Engine” section on page 4-46.</td>
</tr>
</tbody>
</table>

The `proxy incoming` option of the `http`, `https`, `ftp`, and `rtsp` global configuration commands supports up to eight ports per protocol. You can specify up to eight incoming proxy ports on a single command line or on multiple command lines. Proxy-style requests in HTTP, FTP, HTTPS, and RTSP protocols can be received on the same incoming proxy port.

**Note** Both transparent and proxy-style requests can be serviced on the same port.

In this example, the standalone Content Engine is configured to accept HTTP, HTTPS, and FTP-over-HTTP proxy requests directly from client browsers 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-over-http proxy incoming 81 8080 8081
```

In the ACNS 5.3.1 software release, the `ftp` keyword was replaced with the `ftp-over-http` and `ftp-native` keywords to clearly differentiate between FTP-over-HTTP caching and FTP native caching. Consequently, the `ftp proxy incoming` global configuration command was replaced with the `ftp-over-http proxying incoming` and `ftp-native proxy incoming` global configuration commands in the ACNS 5.3.1 software release. The `ftp-native proxy incoming` command, which was added in the ACNS 5.3.1 software release, is used to configure an incoming port for FTP native requests from FTP clients (for example, Reflection X or WS-FTP clients) to support nontransparent FTP native caching. For more information about nontransparent FTP native caching, see the “Configuring Nontransparent FTP Native Caching” section on page 7-42.

The following example shows how to configure an incoming proxy port a standalone Content Engine to listen for incoming WMT traffic on a specific port. These WMT requests are being sent directly to the Content Engine (nontransparent forward proxy) from client Windows Media players that are configured to point directly to the Content Engine. The default WMT port is 1755. Valid port numbers are from 1 to 65535.

```
ContentEngine(config)# wmt port incoming portnumber
```

The following example shows how to configure an incoming proxy port on a standalone Content Engine to listen for incoming RTSP traffic on a specific port. These RTSP requests are being sent directly to the Content Engine (nontransparent forward proxy) from client media players that are configured to point directly to the Content Engine. The default RTSP port is 554. Valid port numbers are from 1 to 65535.

```
ContentEngine(config)# rtsp port incoming portnumber
```
To disable HTTP, HTTPS, FTP, and RTSP incoming proxy services, use the `no protocol proxy incoming` global configuration command (for example, the `no wmt proxy incoming` global configuration command). To add or remove ports in proxy mode, enter a new command that specifies all the ports to be used.

### SSL Tunneling and Nontransparent Caching Deployments

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.

**Tip**

Browsers only initiate HTTPS-over-HTTP requests when they are explicitly configured for a proxy. If a browser is not explicitly configured for a proxy, the browser initiates an HTTP-over-SSL connection itself, and because this is on TCP port 443, the request is only intercepted by the Content Engine if the Content Engine is running the ACNS 5.1.5 software and later releases.

SSL on port 443 uses end-to-end encryption, and any transparent device between the client and the origin server sees nothing more than a stream of random bytes.

### Deploying Caching and Streaming Services in Transparent Mode

In the ACNS 5.x software, standalone Content Engines can handle transparently redirected content requests from a WCCP router and a Layer 4 switch. With transparent redirection, the standalone Content Engine serves as a transparent (invisible) proxy server to web clients or web servers (for example, web servers in a server farm at a company headquarters). When the Content Engine is proxying for web clients, it is a transparent forward proxy server. When it is proxying for a web server, the Content Engine is a transparent reverse proxy server.

### Overview of Transparent Forward Proxy Caching

When you deploy a standalone Content Engine as a transparent cache (a transparent forward proxy server), you place the Content Engine in the middle between web clients and the origin servers; close to the user population. The transparent cache is placed in one of these places, in the path of the network traffic where all egress traffic is guaranteed to pass:

- At the network edge (branch offices)
- At the Internet edge (regional offices or headquarters)

With transparent redirection, web clients request content directly from the source (origin servers). However, these requests are transparently intercepted at a network egress point. A network device (a WCCP router or a Layer 4 Cisco Content Services Switch [CSS] switch) transparently intercepts and redirects these requests to the standalone Content Engine that is functioning as a transparent caching engine for these clients. **Figure 3-2** shows an example of transparent forward proxy caching with a WCCP router.
By supporting WCCP Version 2 or by interoperating with a Layer 4 CSS switch, standalone Content Engines can achieve a basic level of transparency, which includes transparent receipt of content traffic, fault tolerance, and scalable clustering.

Because a Content Engine can be transparent to the client and to network operation, you can easily place standalone Content Engines in several network locations in a hierarchical fashion. For example, an ISP can deploy a standalone Content Engine (Content Engine A) at their main point of access to the Internet so that all of their points of presence (POPs) benefit, because requested content can be available at this main point of access without going through the Internet.

To further improve service to certain web clients, the ISP can deploy a standalone Content Engine (Content Engine B, C, and D) 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 (Content Engine B, C, or D) is unable to fulfill the request from local storage, it makes a normal web request to the origin server. Upstream, this request is redirected to Content Engine A. If the request is fulfilled by Content Engine A, traffic on
Chapter 3      Deployment Scenarios for Standalone Content Engines

Deploying Caching and Streaming Services in Transparent Mode

the main Internet access link is avoided, the origin servers experience lower demand, and the client experiences better network response times. Enterprise networks can apply this hierarchical transparent architecture in the same way.

Although transparent caching deployments require an understanding of the network topology and traffic patterns, there are some significant advantages to deploying transparent mode services on standalone Content Engines:

- No end user configuration—No desktop configuration changes are required.
- Fail-safe operation—Caches are automatically fault-tolerant and fail-safe. Any cache failure does not cause denial of service to the end user.
- Scalability—Cache service can be scaled by deploying multiple caches (cache clusters or hierarchical caches).
- Automatic bypass—Sites that depend on end user authentication or that fail to conform to HTTP standards will automatically bypass a transparent cache.

Transparent Redirection and Forward Proxy Caching with a Layer 4 Switch

When Layer 4 switching is used for transparent redirection, a Layer 4 CSS switch transparently intercepts and redirects content requests to the Content Engine. With transparent interception through a CSS switch, the user is unaware that the request made to an origin web server is redirected to the Content Engine by the Layer 4 CSS switch. The Layer 4 CSS switch can be configured to dynamically analyze the request and determine if the requested content is cacheable or not. If the requested content is not cacheable, the Layer 4 CSS switch sends the request directly to the origin server. If the requested content is cacheable, the Layer 4 CSS switch directs the request to the Content Engine. The Content Engine either returns the requested content if it has a local copy or sends a new request to the origin web server for the content.

When Layer 4 switching is used to redirect content requests transparently to standalone Content Engines, if the TCP SYN packet passes through the Layer 4 CSS switch that has the Layer 4 redirection feature turned on, the packet is diverted to a Content Engine that is attached to the switch. In this case, the Layer 4 CSS switch changes the MAC address of the TCP SYN packet, so that instead of going out to the gateway or the origin server, the MAC address is changed to that of the Content Engine. The Layer 4 CSS switch then sends the packet to the Content Engine. All of these actions occur in the hardware.

The Content Engine must be prepared to accept requests that are sent to it even if the IP address of the packet is not the address of the Content Engine. The Content Engine then handles the TCP SYN packet similarly to how it handles packets redirected through WCCP.

Transparent forward proxy caching with a standalone Content Engine and a Layer 4 CSS switch works as follows:

1. A user (web client) requests a web page from a browser.
2. The Layer 4 CSS switch analyzes the request and determines whether the requested content is cacheable or not. If the requested content is cacheable, the Layer 4 CSS switch transparently redirects the request to a Content Engine.

Note: If all the Content Engines are unavailable in a transparent cache configuration, the Layer 4 CSS switch allows all client requests to progress to the origin web servers.
3. If the Content Engine has the requested content already stored in its local cache, it returns the requested content to the client.

4. If the Content Engine does not have the requested content, the following events occur:
   a. The Content Engine sets up a separate TCP connection to the origin web server to retrieve the content.
   b. The content returns to, and is stored on, the Content Engine.

5. The Content Engine sends the requested content to the web client. Upon subsequent requests for the same content, the Content Engine transparently fulfills the request from its local storage (cache).

**Transparent Redirection and Forward Proxy Caching with a WCCP Router**

A router with the proper configuration is required for transparent caching services. The router must be running a version of Cisco IOS software that supports WCCP Version 1 or Version 2. When caching support is enabled on the router and WCCP support is enabled on the Content Engines, the devices can communicate and deliver the services for which they are configured. 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.

**Tip**

To suspend caching services, you can disable caching support on the router rather than powering off or otherwise disabling individual Content Engines. (For instance, use the `no ip wccp` router command to disable caching.)

Using WCCP, the router transparently redirects requests to the specified TCP ports on the Content Engine rather than to the intended host sites.

WCCP runs on UDP port 2048 within a generic routing encapsulation (GRE) tunnel between the WCCP-enabled router and the Content Engine. 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.

A WCCP Version 1 router can look for TCP as the protocol field in the IP header and for port 80 as the destination port in the TCP header. If the packet meets these criteria, it is redirected to a Content Engine. With WCCP Version 1, web-cached information can only be redirected to a Content Engine 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 port 80, RealAudio, and video.

If a router is configured for WCCP Version 2 as opposed to WCCP Version 1, then the router can be configured to redirect traffic to the Content Engine on TCP ports other than port 80. For example, if you configure the custom-web-cache service (service 98) on the router and Content Engine, the routers can redirect HTTP traffic to Content Engines on a port other than port 80. The Content Engine can then be configured to cache the content from these redirected HTTP requests. This type of caching is called HTTP transparent caching with WCCP.

A transparent request is a request redirected to the Content Engine from a router. 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 hostname, but RTSP requests are the same for both server-style and proxy-style URLs. If a server-style URL is received, only HTTP and RTSP are supported (if the RTSP user agent criteria are met). If a proxy-style URL is received, HTTP, HTTPS, FTP, and RTSP are supported when the respective proxy services are configured.
To configure WCCP as a redirection method, you must perform the following tasks:

1. Enable WCCP redirection on the router, as described in the “Configuring WCCP Services on a Router” section on page 6-27.

2. Enable WCCP on the standalone Content Engine. Configure the specific WCCP services (for example, the web-cache service) that you want the WCCP-enabled router and Content Engine to support.

   For more information on this topic, see the “Configuring Standalone Content Engines for WCCP Transparent Redirection” section on page 6-9.

**Note**

For a complete list of supported WCCP services, see Table B-3.

**Overview of Transparent Reverse Proxy Caching**

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, Content Engines 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. A Content Engine deployed in this manner is called a reverse proxy cache because the Content Engine is operating at the opposite end of the transaction, in front of the origin server.

By having the Content Engine (the reverse proxy server) transparently handle inbound requests for content instead of having the origin servers (the servers in the server farms) handle these requests, web traffic is significantly reduced. Reverse proxy servers are an effective method for scaling server farms.

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 DNS resolution of a domain name. To a client, the reverse proxy server appears like a web server.

The ACNS 5.x software provides reverse proxy caching by allowing traffic redirection or interception to be performed by two types of devices: a WCCP Version 2-enabled router or a Layer 4 CSS switch. Figure 3-3 shows a typical reverse proxy caching deployment with a WCCP-enabled router. In this type of deployment, the Content Engine interoperates with the WCCP Version 2-enabled router to bring the reverse-proxy service (service 99) within the web server environment. The Content Engine is deployed in front of a web server farm. Unlike transparent and nontransparent forward proxy servers, the reverse proxy server proxies requests on behalf of a server farm, and it only caches content from servers in the server farm.
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.

For more information about implementing reverse proxy caching through a WCCP router or Layer 4 CSS switch, respectively, see the “Example 1—Deploying Reverse Proxy Caching with WCCP Transparent Redirection” section on page 3-12, and the “Example 2—Deploying Reverse Proxy Caching with Layer 4 Switching” section on page 3-13.

In Figure 3-3, the router interface connected to the Internet has an IP address of 192.168.1.1. All HTTP requests destined for web server B are routed to the router interface at 172.16.21.1. Upon receiving the HTTP request at this interface, the router transparently intercepts and redirects the request to the Content Engine that has an IP address of 172.16.20.23. Thus, the Content Engine is logically in front of web server B, offloading web server HTTP traffic. Web clients who are requesting content from the origin server receive the static web pages from the Content Engine that is acting in reverse proxy mode. This frees up the back end infrastructure from processing this HTTP traffic.

The following are some significant advantages to deploying reverse proxy caching on standalone Content Engines:

- Provides an alternative to web server expansion by offloading the processing of static images from the server farm.
- Provides a possible way of replicating content to geographically dispersed areas by deploying Content Engines in these areas.
- Does not require any client configuration changes (you do not need to configure the client browsers to point to the Content Engine that is functioning as the reverse proxy server).
Example 1—Deploying Reverse Proxy Caching with WCCP Transparent Redirection

This example shows how to deploy the reverse proxy caching service using WCCP transparent redirection (as opposed to Layer 4 switching).

Step 1  Enable WCCP Version 2 on the standalone Content Engine. (WCCP Version 1 does not support the reverse-proxy service.)

```
ContentEngine(config)# wccp version 2
```

Step 2  Configure a router list.

```
ContentEngine(config)# wccp router-list 1 172.16.20.1
```

Step 3  Instruct the Content Engine to run the reverse-proxy service (service 99).

```
ContentEngine(config)# wccp reverse-proxy router-list-num 1
```

Step 4  Exit global configuration mode.

```
ContentEngine(config)# exit
```

Step 5  Instruct the router to run the reverse-proxy service (service 99).

```
Router(config)# ip wccp 99
```

Step 6  Configure the reverse-proxy service with output redirection facing the original servers by performing these steps:

   a. Specify which router interface to configure. In this example, Ethernet 0/1 is the router interface to the web server.

```
Router(config)# interface Ethernet 0/1
```

   b. Instruct the router to redirect TCP port 80 traffic bound for the specified interface to Content Engines that accept reverse proxy service. In this example, there is only one router.

```
Router(config-if)# ip wccp 99 redirect out
```

Step 7  Configure the reverse-proxy service with input redirection facing the standalone Content Engine by performing these steps:

   a. Specify which router interface to configure. In this example, s0/0 is the router interface to the Internet.

```
Router(config)# interface s0/0
```

   b. Instruct the router to redirect TCP port 80 traffic received on the specified interface to Content Engines that accept reverse proxy service.

```
Router(config-if)# ip wccp 99 redirect in
```

Step 8  Exit interface configuration mode.

```
Router(config-if)# exit
```
Example 2—Deploying Reverse Proxy Caching with Layer 4 Switching

With reverse proxy caching that is based on a Layer 4 switch, the standalone Content Engine interoperates with a Layer 4 CSS switch to bring reverse proxy service within the web server environment. In this case, a set of Content Engines use their local caches to accelerate the actual web server. The Layer 4 CSS switch is a load-balancing switch that has a virtual IP address (for example, 200.200.200.1). This virtual IP address is the IP address of the web server that is seen by the rest of the world (web clients that are requesting information from that server). Client requests arrive at the Layer 4 switch first; the switch has Layer 4 redirection capabilities and it redirects the requests to the Content Engine (reverse proxy server) that is attached to the Layer 4 switch. If the request is a cache hit, then the Content Engine services the request. If the request is a cache miss, the Content Engine sends the request to the web server at the back end, caches the result, and sends the requested content to the client.

In reverse proxy caching cases, the Content Engines are transparent to the rest of the world and the clients typically are not aware of the presence of the Content Engines (reverse proxy servers).

Note: The sample solution shown here works only with particular web servers and in particular configurations.

This example shows how to configure reverse proxy caching based on a Layer 4 CSS switch (CS150) and a standalone Content Engine (ce1).

Step 1
Disable the load bypass feature to ensure that the Content Engine serves requests if it can.

cel(config)# no bypass load enable

Step 2
Configure the Content Engine to accept traffic redirection using the Layer 4 switch.

cel(config)# http l4-switch enable

Step 3
Configure the Layer 4 switch for reverse proxy caching.
The Layer 4 switch must be in configuration mode in order to initiate configuration changes.

Note: See the CSS Advanced Configuration Guide for more information regarding caching with a Layer 4 CSS switch.

a. Specify the owner of the CSS switch.

CS150(config)# owner cisco

b. Create a reverse proxy rule for the current owner.

CS150(config-owner[cisco])# content RPCRule
Create content RPCRule>, [y/n]:y

c. Add services to the reverse proxy rule. In this case, the standalone Content Engine (ce1) is added as a service to the reverse proxy rule.

CS150(config-owner-content[RPCRule])# add service ce1

d. Assign a virtual IP address to the CSS switch.

CS150(config-owner-content[cisco-RPCRule])# vip address 200.200.200.1
Deploying Streaming Media Services on Standalone Content Engines

Step 4
Exit configuration mode on the CSS switch.

CS150(config-owner-content[cisco-RPCRule])# exit

Using Advanced Transparent Caching Features

One of the fundamental principles of transparent network caching is that the Content Engine 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 ACNS software uses a WCCP-enabled router and various advanced techniques to ensure that the Content Engine remains transparent, even if client browsers are nonoperational or web servers are not HTTP-compliant. For more information on this topic, see Chapter 15, “Configuring Advanced Transparent Caching Features on Standalone Content Engines.”

Deploying Streaming Media Services on Standalone Content Engines

Streaming is a technology that allows content to be accessed or viewed before all the media packets have been received, with caching, the content must be received in its entirety before it can be accessed. Streaming media can be delivered as live content or as on-demand content, such as video on demand (VOD).

Cisco ACNS software supports several types of streaming media solutions, including the Windows Media Technologies (WMT) solution from Microsoft Corporation and the RealMedia solution from RealNetworks, Inc. See Table 1-3 for a list of supported streaming media solutions.

For information about how to deploy RealMedia streaming services on standalone Content Engines, see Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.” For information about how to deploy WMT streaming services on standalone Content Engines, see Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”

For information about how to configure streaming media services for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Deploying Filtering and Access Control Services

After you configure caching and streaming services on a standalone Content Engine, you can configure specific content services (for example, rules processing and URL filtering). This section provides an example of how to configure a standalone Content Engine to control user Internet access. In this example, a company wants to deploy a standalone Content Engine as a caching engine in its enterprise. The company also has the following special requirements for processing content requests:

- The standalone Content Engine should only allow Internet access to users on selective subnets, and should limit their Internet access to specific websites.
- The standalone Content Engine should block specific users on the permitted subnets from accessing the permitted websites.
- The standalone Content Engine should block Internet access to all other users, and display a custom message informing users that their request to access the website has been denied.
- Because there are only a small number of sites and users, the company does not want to implement a third-party URL filtering solution (for example, the SmartFilter product).

This example shows how to implement a solution that meets the above requirements with a standalone Content Engine that is running the ACNS 5.x software.

**Step 1** Enable rules processing on the standalone Content Engine.

```
ContentEngine(config) # rule enable
```

**Step 2** Block selective users who belong to the permitted subnet from accessing the normally permitted sites. In this case, the Content Engine blocks users who belong to the subnet 192.168.1.50 and the domain foo.com from accessing the normally permitted sites.

```
ContentEngine(config) # rule action block pattern-list 2 protocol all
ContentEngine(config) # rule pattern-list 2 group-type and
ContentEngine(config) # rule pattern-list 2 src-ip 192.168.1.50 255.255.255.0
ContentEngine(config) # rule pattern-list 2 domain .*foo\.com
```

**Step 3** Define a custom message that you want the Content Engine to display to a blocked user.

a. Create a separate directory under local1 or local2 for holding the custom message file.

b. Create an HTML file named block.html that contains the blocking message. Make sure to copy all embedded graphics associated with the custom message HTML page to the same directory that contains the block.html file. The following is an example of a block.html file:

```html
<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>
</BLINK>
<FONT COLOR="#0080FF">Blocked Page</FONT>
</I></B></CENTER>
</H1>
</p>
```
Step 4  Configure the Content Engine to send blocked users the custom message. Specify the directory (dirname) the block.html file.

```
ContentEngine(config)# url-filter http custom-message dirname
```

In this example, the block.html file displays the following custom message when the Content Engine intercepts a request to the blocked site:

```
This page is blocked by the Content Engine.
```

Step 5  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 that are in a goodurl.lst file. The use of URL lists applies to requests in HTTP, HTTPS, and FTP format as well as streaming media protocols such as MMS-over-HTTP and RTSP. The following example shows how to permit specific HTTP URLs to the exclusion of all other URLs:

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

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

c. 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 good lists, for example, /local1/filtered_urls.

Step 6  Point the Content Engine to the goodurl.lst filename.

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

Step 7  Configure the Content Engine to actively permit access to only the good URLs.

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

For more information about how to configure URL filtering and rules on standalone Content Engines, see Chapter 11, “Configuring Content Preloading and URL Filtering on Standalone Content Engines” and Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”
PART 2

Basic Configuration for Standalone Content Engines
Getting Started

This chapter provides an overview of how to configure, monitor, and troubleshoot standalone Content Engines that are running the ACNS 5.4.1 software and later releases. It also describes how to use the ACNS software Setup utility to configure the general settings (device network settings and disk configuration) and a set of commonly used caching services (listed in Table 4-2) on a standalone Content Engine.

This chapter contains the following sections:

- **Overview of Configuring, Monitoring, and Troubleshooting Standalone Content Engines**, page 4-2
- **Configuring a Basic Configuration on Standalone Content Engines with the Setup Utility**, page 4-10
- **Configuring Client Browsers and Media Players for Direct Proxy Routing**, page 4-35
- **Configuring WCCP Routers for Transparent Redirection**, page 4-47
- **Verifying the Basic Configuration**, page 4-47
- **Modifying the Basic Configuration Through the Setup Utility**, page 4-50
- **Logging in to Standalone Content Engines**, page 4-50

The term *standalone Content Engines* is used throughout this guide to refer to Content Engines that ACNS administrators have intentionally not registered with the Content Distribution Manager so that they can configure, manage, and monitor these Content Engines as standalone devices. Multiple standalone Content Engines can be deployed (for example, you can deploy clusters of standalone Content Engines).

After you have done a basic configuration on a standalone Content Engine, you can perform other basic tasks such as setting the system clock, managing login accounts, and managing and monitoring disks. For more information on this topic, see Chapter 5, “Performing Other Basic Tasks for Standalone Content Engines.”

---

For complete syntax and usage information for the CLI commands used in this chapter, see the *Cisco ACNS Software Command Reference, Release 5.5* publication. For information about how to configure Content Engines that are registered with a Content Distribution Manager, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*.
Overview of Configuring, Monitoring, and Troubleshooting Standalone Content Engines

This section provides an overview of how to configure, monitor, and troubleshoot standalone Content Engines as caching and streaming engines, and contains the following sections:

- Flowcharts of Configuring, Monitoring, and Troubleshooting Standalone Content Engines, page 4-2
- Checklist for Configuring, Monitoring, and Troubleshooting Standalone Content Engines, page 4-7

Flowcharts of Configuring, Monitoring, and Troubleshooting Standalone Content Engines

Figure 4-1 shows a high-level view of a typical workflow for configuring, monitoring, and troubleshooting a standalone Content Engine. Table 4-1 provides a checklist of tasks for completing the workflow that is shown in Figure 4-1.

Note As the legend in Figure 4-1 indicates, more detailed flowcharts are provided for configuring conventional caching services (Figure 4-2) and RealMedia streaming and caching services (Figure 4-3).
Figure 4-1  High-Level View of Configuring, Monitoring, and Troubleshooting Standalone Content Engines

Begin configuration of a standalone Content Engine

Choose one of the following addressing schemes to initially configure this Content Engine:
- Static IP address and network mask
- Dynamically assigned IP address

Power up the Content Engine and open a console connection on one of its serial ports

Use the interactive Setup utility or CLI commands to configure these basic settings:
- Device network settings
- Disk configuration

Which services will be deployed on this Content Engine?
- Conventional caching services
  - DNS caching
  - HTTP caching
  - FTP caching
  - HTTPS caching (rare)

Streaming media services
- WMT streaming and caching
- RTSP streaming and caching

Configure one or more of the following routing methods to direct content requests to this standalone Content Engine:
- Direct proxy routing (nontransparent)
- WCCP routing or Layer 4 switching (transparent)

Basic configuration of standalone Content Engine is complete

Conventional caching services
Configure selected conventional caching services
Configure selected WMT streaming and caching services
Configure selected RTSP streaming and caching services

Streaming media services
Configure WMT streaming and caching
Configure RTSP streaming and caching

Legend:
- Actions within box area can be performed in any order
- Action within box area is broken down into a more detailed flowchart

Monitor and troubleshoot
- Monitor Content Engine
- Monitor transactions
- Use logs and traceroute for troubleshooting

Perform advanced configuration on the Content Engine
- Set up additional network interfaces
- Configure bandwidth for interfaces and content services
- Set up login authentication and authorization
- Configure system accounting with TACACS+
- Configure IP ACLs
- View or modify TCP stack parameters
- View or modify system logging settings
Overview of Configuring, Monitoring, and Troubleshooting Standalone Content Engines

Chapter 4      Getting Started

Figure 4-2   Detailed View of Configuring Conventional Caching Services for Standalone Content Engines

Begin configuration of conventional caching services for a standalone Content Engine

Which routing methods will be used to direct client requests to this standalone Content Engine?

WCCP routing or Layer 4 switching (transparent redirection)  Direct proxy routing (nontransparent)

Which transparent mode conventional caching services will be deployed?

- HTTP reverse proxy caching
- Native FTP caching
- HTTPS transparent caching
- HTTP transparent caching
- DNS caching

Which nontransparent (proxy) mode conventional caching services will be deployed?

- HTTP forward proxy caching
- FTP-over-HTTP caching
- HTTPS proxy caching

Configure one or more of the following transparent redirection methods to direct content requests to this standalone Content Engine (transparent proxy server)

- On this Content Engine, specify the proxy incoming ports for each selected caching service and enable the service
- Point client browsers directly to this standalone Content Engine (nontransparent forward proxy server)
  - Manually configure the client browsers or
  - Use the proxy autoconfiguration feature (.pac file)

WCCP  Layer 4 switching

Configure redirection through WCCP
- Configure on WCCP routers that will support the caching service
- Configure on this Content Engine

Configure redirection through Layer 4 switching
- Configure on Layer 4 switch
- Configure on this Content Engine

Configuration of conventional caching services for a standalone Content Engine is complete.
This chapter describes how to use the Setup utility to configure the following three commonly used conventional caching services on standalone Content Engines: HTTP reverse proxy caching, HTTP transparent caching using WCCP Version 2, and HTTP forward proxy caching. For information about how to use the Content Engine CLI method (instead of the Setup utility) to configure these three services as well as other conventional caching services (for example, DNS caching and FTP caching), see Chapter 7, “Configuring Conventional Caching Services for Standalone Content Engines.”

This chapter describes how to use the Setup utility to configure the following two commonly used WMT MMS caching services on standalone Content Engines: WMT transparent caching and WMT proxy caching. For information about how to use the Content Engine CLI (instead of the Setup utility) to configure these caching services, other WMT RTSP services on a standalone Content Engine, see Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”
Begin configuration of RealMedia services for a standalone Content Engine

Use the Setup utility or Content Engine CLI to enable RealProxy on the standalone Content Engine
- Accept the RealProxy license agreement
- Specify a RealProxy license key (accept the evaluation RealProxy license key, or enter your permanent key)
- Turn on the RealProxy feature on this Content Engine

Use the Content Engine CLI to specify the following settings for the RTSP gateway (the single point of entry for RTSP messages) that runs on the standalone Content Engine and is automatically enabled
- Specify the IP address of the RTSP gateway if the Content Engine is behind a NAT-enabled router (required)
- Change the default basic settings, such as the RTSP incoming port number (optional)
- Change the default advanced settings, such as the maximum request rate (optional)

Configure one or more of the following routing methods to direct RealMedia client requests to the RTSP gateway on the standalone Content Engine

**WCCP routing or Layer 4 switching**
- WCCP routing
- Layer 4 switching

**Direct proxy routing (nontransparent)**

**Configure RTSP redirection through WCCP (service 80)**
- Configure on WCCP routers that will support this media service
- Configure on this Content Engine

**Configure RTSP redirection through Layer 4 switching**
- Configure on Layer 4 switch
- Configure on this Content Engine

**Manually configure RealMedia players to point directly to this Content Engine**
- (nontransparent forward proxy server)

RealMedia transparent caching (caching VOD files) and live splitting with RealProxy

RealMedia proxy caching (caching VOD files) and live splitting with RealProxy

Use the RealSystem Administrator GUI to configure RealProxy on this Content Engine
- (for example, configure live splitting)

Configuration of RealMedia services for a standalone Content Engine is complete
Overview of Configuring, Monitoring, and Troubleshooting Standalone Content Engines

This chapter describes how to use the Setup utility to configure the following two commonly used RealMedia caching services on standalone Content Engines: RealMedia transparent caching and RealMedia proxy caching. For information about how to use the Content Engine CLI (instead of the Setup utility) to configure these caching services or other RealMedia services (for example, RealProxy live splitting) on standalone Content Engines, see Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”

Checklist for Configuring, Monitoring, and Troubleshooting Standalone Content Engines

Table 4-1 is a checklist of tasks for configuring, monitoring, and troubleshooting standalone Content Engines that are running the ACNS 5.4.1 software and later releases.

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
</table>
| Start basic configuration | The two supported addressing schemes are mutually exclusive:  
  - Manually specify a static IP address and network mask  
  - Dynamically assign an IP address using the interface-level DHCP addressing scheme  
  See the “Deciding the Addressing Scheme for Standalone Content Engines” section on page 4-16. |
| 1. Decide which addressing scheme will be used to initially configure this standalone Content Engine. | This chapter describes how to use the Setup utility to expedite the basic configuration of standalone Content Engines. See the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.  
  A brief description of how to use the CLI method to configure general settings is provided in the “Using the CLI Command Method to Configure General Settings for Standalone Content Engines” section on page 4-18. For detailed information about the CLI commands used to configure general settings, see the Cisco ACNS Software Command Reference, Release 5.5 publication.  
  For information about how to use the CLI method to configure or modify one or more commonly used caching services and numerous other services (for example, DNS caching, FTP caching, WMT streaming, and RealMedia streaming services) running on standalone Content Engines, see the following chapters in this guide:  
  - Chapter 7, “Configuring Conventional Caching Services for Standalone Content Engines”  
  - Chapter 8, “Configuring RealMedia Services on Standalone Content Engines”  
  - Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines” |
Table 4-1 Checklist for Configuring, Monitoring, and Troubleshooting Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Power up this Content Engine and open a console connection on one of its serial ports.</td>
<td>After you physically install the hardware and power up the Content Engine, you can access the ACNS software (Setup utility or CLI commands) to perform a basic configuration of this standalone (unregistered) Content Engine (instead of a Content Engine that will be registered with a Content Distribution Manager).</td>
</tr>
<tr>
<td>4. Use the Setup utility or CLI commands to configure the following general settings on this standalone Content Engine:</td>
<td>To use the Setup utility to configure the general settings, see the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.</td>
</tr>
<tr>
<td>– Device network settings</td>
<td>To use the CLI command method to configure the general settings, see the “Using the CLI Command Method to Configure General Settings for Standalone Content Engines” section on page 4-18.</td>
</tr>
<tr>
<td>– Disk configuration</td>
<td></td>
</tr>
<tr>
<td>5. Choose which services will be deployed on this standalone Content Engine.</td>
<td>See the “Overview of Configuring Conventional Caching Services” section on page 7-2.</td>
</tr>
<tr>
<td>– Conventional caching services (DNS, HTTP, FTP, and HTTPS caching)</td>
<td>See Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”</td>
</tr>
<tr>
<td>– RealMedia streaming and caching services</td>
<td>See Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”</td>
</tr>
<tr>
<td>– WMT streaming and caching services</td>
<td></td>
</tr>
<tr>
<td>6. Configure one or more of the following routing methods to direct content requests to this standalone Content Engine:</td>
<td>For direct proxy routing, see the “Configuring Client Browsers and Media Players for Direct Proxy Routing” section on page 4-35.</td>
</tr>
<tr>
<td>– Direct proxy routing (nontransparent)</td>
<td>For WCCP routing, see the “Configuring WCCP Services on a Router” section on page 6-27.</td>
</tr>
<tr>
<td>– Transparent redirection (WCCP routing or Layer 4 switching)</td>
<td>For Layer 4 switching, see the “Configuring Layer 4 Switching as a Redirection Method” section on page 6-50.</td>
</tr>
<tr>
<td>7. If direct proxy routing is to be used, is a *.pac file to be used?</td>
<td>• If no, then manually configure each client browser to point directly to the standalone Content Engine as a direct proxy server, as described in the “Manually Pointing Client Browsers to a Standalone Content Engine” section on page 4-42.</td>
</tr>
<tr>
<td></td>
<td>• If yes, then configure the standalone Content Engine and the client browsers to use a proxy autoconfiguration (PAC) file, as described in the “Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine” section on page 4-37.</td>
</tr>
<tr>
<td>8. Configure the chosen caching and streaming services on this standalone Content Engine.</td>
<td>The Setup utility allows you to configure a set of commonly used caching services (listed in Table 4-2) on a standalone Content Engine. We recommend that you use this utility to configure one or more of these caching services on your Content Engine. This allows you to get your Content Engine up and running a basic set of caching services. For more information, see the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.</td>
</tr>
<tr>
<td>9. Verify the basic configuration.</td>
<td>Now that the basic configuration is completed, verify that these caching services are working properly. See the “Verifying the Basic Configuration” section on page 4-47.</td>
</tr>
</tbody>
</table>
### Table 4-1 Checklist for Configuring, Monitoring, and Troubleshooting Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
</table>
| 10. | You can now do any of the following tasks:  
- Configure content services.  
- Perform advanced configuration on this Content Engine.  
- Monitor and troubleshoot.  
See tasks 11 through 23 below in this table. |
| Configure content services (optional) | After configuring caching and streaming services on the standalone Content Engine, you can configure such content services as access control, URL filtering, ICAP, and rules. |
| 11. | Decide if end user access to the Internet is to be controlled (access control for HTTP, HTTPS, and FTP-over-HTTP requests).  
- If no, then go to task 12.  
- If yes, then configure authentication and authorization, as described in Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.” |
| 12. | Decide if URL filtering is to be used.  
- If no, then go to task 13.  
- If yes, then configure URL filtering for HTTP, HTTPS, and FTP requests, as described in Chapter 11, “Configuring Content Preloading and URL Filtering on Standalone Content Engines.” |
| 13. | Determine whether there is an external ICAP server.  
- If no, then go to task 14.  
- If yes, then configure the Internet Content Adaptation Protocol (ICAP) for HTTP and FTP-over-HTTP requests, as described in Chapter 12, “Configuring ICAP on Standalone Content Engines.” |
| 14. | Determine if there are any special requirements for processing content requests.  
- If no, then go to task 15.  
- If yes, configure rules for HTTP, HTTPS, FTP-over-HTTP, WMT, and RTSP requests, as described in Chapter 13, “Configuring the Rules Template on Standalone Content Engines.” |
| Perform advanced configuration tasks (optional) |  
| 15. | Configure advanced transparent caching features (for example, traffic bypass, overload bypass, flow protection, and IP spoofing).  
Chapter 15, “Configuring Advanced Transparent Caching Features on Standalone Content Engines” |
| 16. | Set up additional network interfaces on the standalone Content Engine.  
Chapter 16, “Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines” |
| 17. | Configure bandwidth for interfaces and content services on this standalone Content Engine.  
Chapter 16, “Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines” |
| 18. | Set up login authentication and authorization on this standalone Content Engine.  
Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines” |
| 19. | Configure this standalone Content Engine for system accounting with TACACS+.  
Chapter 18, “Configuring AAA Accounting on Standalone Content Engines” |
Configuring a Basic Configuration on Standalone Content Engines with the Setup Utility

This section provides an overview of the Setup utility and describes how to use this tool to configure a basic configuration on a standalone Content Engine in either of the situations:

- Case 1—The Content Engine is being booted up for the first time (for example, the Content Engine was purchased with the ACNS 5.4 software), and you want to use the Setup utility to configure the basic configuration settings (the general settings [device network settings and disk configuration], and a set of commonly used caching services [listed in Table 4-2]).

- Case 2—The device is a standalone Content Engine with some basic configuration (for example, the Content Engine was upgraded to the ACNS 5.4 software, and already has device network settings, disk configuration, and HTTP proxy caching configured). You want to use the Setup utility to complete the basic configuration of this standalone Content Engine (for example, configure some of the other commonly used caching services that are not yet configured).

In Case 1, the Setup utility is automatically launched when you initially boot up a device. Completing the initial basic configuration in this situation involves these tasks:

- Deciding the Addressing Scheme for Standalone Content Engines, page 4-16
- Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine, page 4-21

In Case 2, you manually launch the Setup utility with the `setup` privileged EXEC command. For more information on this topic, see the “Manually Launching the Setup Utility” section on page 4-19.

After completing this basic configuration, you must configure the client browsers and media players (see “Configuring Client Browsers and Media Players for Direct Proxy Routing”) for direct proxy routing, and configure the WCCP routers for transparent redirection (see “Configuring WCCP Routers for Transparent Redirection”). After verifying that this basic configuration is working properly, you can use

---

**Table 4-1** Checklist for Configuring, Monitoring, and Troubleshooting Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.</td>
<td>Configure IP access control lists (ACLs) on this standalone Content Engine.</td>
</tr>
<tr>
<td>21.</td>
<td>View or modify TCP stack parameters for this standalone Content Engine.</td>
</tr>
<tr>
<td>22.</td>
<td>View or modify the system logging settings for this standalone Content Engine.</td>
</tr>
<tr>
<td>Monitor and troubleshoot</td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>Monitor this standalone Content Engine with SNMP, the ACNS software alarms, and the ACNS software logs.</td>
</tr>
<tr>
<td>24.</td>
<td>Use the traceroute and the other supported diagnostic tools for troubleshooting.</td>
</tr>
</tbody>
</table>
the CLI commands or Content Engine GUI to configure additional caching services (for example, DNS caching, FTP caching, and HTTPS caching), streaming services (WMT streaming and RTSP streaming), or content services. You can also perform advanced configuration or monitor this Content Engine.

**Note**
Throughout the rest of this chapter the term *WCCP Version 2-enabled router* denotes a router that is running WCCP Version 2.

### Commonly Used Caching Services Configurable Through the Setup Utility

**Table 4-2** lists the commonly used caching services that you can quickly configure on a standalone Content Engine through the Setup utility.

<table>
<thead>
<tr>
<th>Caching Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP forward proxy caching</td>
<td>The standalone Content Engine functions as a nontransparent forward proxy server for HTTP requests. After receiving an HTTP request directly from a client browser, the Content Engine retrieves and caches the requested content if it is not already stored in its local cache, and sends the requested content to the requester (client browser).</td>
</tr>
<tr>
<td>HTTP transparent caching</td>
<td>The standalone Content Engine functions as a transparent proxy server for HTTP requests. After receiving a redirected HTTP request, the Content Engine retrieves and caches the requested content if it is not already stored in its local cache, and sends the requested content to the requester (client browser). With the Setup utility, you can configure the Content Engine to accept redirected HTTP requests from a WCCP Version 2-enabled router. With the Content Engine CLI, you can configure the Content Engine to accept redirected HTTP requests from a WCCP Version 2-enabled router or a Layer 4 switch.</td>
</tr>
<tr>
<td>HTTP reverse proxy caching</td>
<td>The standalone Content Engine functions as a transparent proxy server for specific web servers (for example, web servers in a web server farm) as opposed to acting as a proxy for end users (web clients). After receiving a redirected reverse proxy request, the Content Engine retrieves and caches the requested content if it is not already stored in its local cache, and sends the requested content to the requester (client browser). With the Setup utility, you can configure the Content Engine to accept redirected reverse proxy requests from a WCCP Version 2-enabled router. With the Content Engine CLI, you can configure the Content Engine to accept redirected reverse proxy requests from a WCCP Version 2-enabled router or a Layer 4 switch.</td>
</tr>
<tr>
<td>WMT proxy caching</td>
<td>The standalone Content Engine functions as a nontransparent proxy server for end users who are using Windows Media Player to request WMT content. After receiving a WMT request directly from a client Windows Media Player, the Content Engine retrieves the requested content if it is not already stored in its local cache, stores a copy locally whenever possible, and sends the requested content to the requester (client Windows Media player).</td>
</tr>
<tr>
<td>WMT transparent caching</td>
<td>The standalone Content Engine functions as a transparent proxy server for end users who are using Windows Media player to request content. After receiving a transparently redirected WMT request, the Content Engine retrieves the requested content if it is not already stored in its local cache, stores a copy locally whenever possible, and sends the requested content to the requester (client Windows Media player). With the Setup utility, you can configure the Content Engine to accept redirected WMT requests from a WCCP Version 2-enabled router. With the Content Engine CLI, you can configure the Content Engine to accept redirected WMT requests from a WCCP Version 2-enabled router or a Layer 4 switch.</td>
</tr>
</tbody>
</table>
Using the Setup Utility

The following are some important points when using the Setup utility:

- When the Setup utility is invoked on a Content Engine that supports device mode changes (for example, the CE-565 or the CE-7306), you are prompted to specify the device mode for that particular device. When prompted, press Enter or enter CE to specify the Content Engine device mode for this standalone Content Engine.

What is the mode of the device (CE/CR/CDM/PM) [CE]: CE

When prompted, enter no to specify that this Content Engine is not going to be managed by a Content Distribution Manager.

Is this CE going to be managed by a CDM (Content Distribution Manager) (y/n) [y]: no

- When a series of basic configuration questions appears, press Enter or enter y to indicate that you want to configure a particular caching service on this standalone Content Engine. (See Table 4-2 for a description of these services).

- After you respond to the series of basic configuration questions, a menu-based interface appears.

```
| +-----------------------------------+ |
| | Main Menu                        | |
| +-----------------------------------+ |
| | -General Settings :Incomplete     | |
| | Caching Related Configurations :Incomplete | |
| | Print Configuration              | |
| | Exit (e)                         | |
| +-----------------------------------+ |
```

Configure network settings, disks, etc
Press '?' to see why this item is incomplete

For a list of the Setup utility menu options, see Table 4-3.

### Table 4-2  Commonly Used Caching Services Configurable Through the Setup Utility (continued)

<table>
<thead>
<tr>
<th>Caching Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealMedia proxy caching</td>
<td>The standalone Content Engine functions as a nontransparent proxy server for end users who are using a RealMedia player to request RTSP content. After receiving an RTSP request directly from a RealMedia player, the Content Engine retrieves the requested content if it is not already stored in its local cache, stores a copy locally whenever possible, and sends the requested content to the requester (the RealMedia player).</td>
</tr>
<tr>
<td>RealMedia transparent caching</td>
<td>The standalone Content Engine functions as a transparent proxy server for end users who are using a RealMedia player to request content. After receiving a redirected RTSP request, the Content Engine retrieves requested content, stores a copy locally whenever possible, and sends requested content to the requester (the RealMedia player). With the Setup utility, you can configure the Content Engine to accept redirected RTSP requests from a WCCP Version 2-enabled router. With the Content Engine CLI, you can configure the Content Engine to accept redirected RTSP requests from a WCCP Version 2-enabled router or a Layer 4 switch.</td>
</tr>
</tbody>
</table>
When you are prompted for a particular configuration setting, the default value is displayed. Press Enter to select the default value. For example, press Enter to specify that you want to use the default WCCP router (default gateway that has the IP address of 10.0.1.1) when prompted as follows:

Please enter the IP addresses of WCCP routers [10.0.1.1]:

After you specify whether you want to configure a particular setting, the corresponding menu option is marked as “Complete” or “Incomplete.” This helps you track which basic configuration settings you have configured and which ones still need to be configured on this standalone Content Engine.

When you configure a particular setting by choosing a menu option in the Setup utility, the corresponding CLI command is configured. (See Table 4-3.) After you use the Setup utility to specify a specific basic configuration setting, a list of configured CLI commands appears (see example). When you are asked if you want to save this configuration, press Enter to save the displayed configuration.

NOTE: Please remember to configure web-cache service on the router.

Based on the input, the following CLIs will be configured:

```
wccp router-list 1 10.0.1.1
wccp version 2
wccp web-cache router-list 1
```

Do you accept these configs (y/n) [y]:

To display a list of constructed CLI commands at any time during a Setup utility session, choose the Print Configurations option from the Setup utility menu.

After you specify a configuration setting, the Setup utility reports any dependencies or incompatibility between the specified options. For example, if you have enabled any of the streaming caching services, then you are informed that you should allocate disk space for the media file system (mediafs) when you configure the disk.

If any failure occurs when the Setup utility is applying the specified configuration settings (for example, the disk configurations and the corresponding CLI command), the Setup utility displays a message indicating which specific setting could not be applied. Error messages are also written to /local1/errorlog/setup_{clidisk}_config_error.

A basic configuration (see example) is constructed based on the information that you specify through the Setup utility. The following is an example of a basic configuration for a standalone Content Engine that has all seven of the commonly used caching services configured:

Here is the current profile of this device

<table>
<thead>
<tr>
<th>CDN device</th>
<th>HTTP Proxy Caching</th>
<th>HTTP Transparent Caching</th>
<th>HTTP Reverse Proxy Caching</th>
<th>WMT Proxy Caching</th>
<th>WMT Transparent Caching</th>
<th>Real Media Proxy Caching</th>
<th>Real Media Transparent Caching</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Do you want to change this (y/n) [n]:

Press the ESC key at any time to quit this session

This basic configuration is cached. A copy of the configurations generated through the Setup utility is stored on disk (/local/local1/setup_gen_config.txt).

For information about how to launch the Setup utility, see the “Launching the Setup Utility” section on page 4-19.
**Setup Utility Menu Options and Corresponding CLI Commands**

The menu structure of the Setup utility is hierarchical. For example, after you choose the General Settings option from the main menu, the General Settings submenu appears. As the following example shows, your current location in the menu structure is displayed after the menu options.

```
+---------------------------------------------+
¦              General Settings               ¦
+---------------------------------------------¦
¦->  Network Configurations :Incomplete       ¦
¦    Disk Configurations    :Incomplete       ¦
¦    Print Configuration                      ¦
¦    Previous Menu (p)                        ¦
¦    Main Menu (m)                            ¦
¦    Exit (e)                                 ¦
+---------------------------------------------+
```

Table 4-3 lists the Setup utility menu option and the corresponding Content Engine CLI command. For detailed descriptions of the CLI commands you can use to configure network settings and disk configuration, see the *Cisco ACNS Software Command Reference, Release 5.4* publication. Information about how to use the CLI method (instead of the Setup utility) to configure any of the commonly used caching services (listed in Table 4-2) and other features are provided in subsequent chapters of this guide.

**Table 4-3  Setup Utility Menu Options and Corresponding CLI Commands for Standalone Content Engines**

<table>
<thead>
<tr>
<th>Setup Utility Menu Option</th>
<th>Content Engine CLI Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Network Configuration</td>
<td>`ip address {ip address netmask</td>
</tr>
<tr>
<td></td>
<td><code>ip default-gateway</code></td>
</tr>
<tr>
<td></td>
<td><code>hostname</code></td>
</tr>
<tr>
<td></td>
<td><code>ip name-servers</code></td>
</tr>
<tr>
<td></td>
<td><code>ip domain-name</code></td>
</tr>
<tr>
<td>Disk Configuration</td>
<td>`disk config sysfs {remaining</td>
</tr>
<tr>
<td></td>
<td>`[cfs {remaining</td>
</tr>
<tr>
<td></td>
<td>`[mediafs {remaining</td>
</tr>
<tr>
<td>Caching-Related Configurations</td>
<td></td>
</tr>
<tr>
<td>HTTP proxy caching</td>
<td><code>http proxy incoming</code></td>
</tr>
<tr>
<td>HTTP transparent caching</td>
<td><code>wccp router list</code></td>
</tr>
<tr>
<td></td>
<td><code>wccp web-cache router-list</code></td>
</tr>
<tr>
<td></td>
<td><code>wccp version 2</code></td>
</tr>
<tr>
<td>HTTP reverse proxy caching</td>
<td><code>wccp router list</code></td>
</tr>
<tr>
<td></td>
<td><code>wccp reverse-proxy router-list</code></td>
</tr>
<tr>
<td></td>
<td><code>wccp version 2</code></td>
</tr>
</tbody>
</table>
### Table 4-3  Setup Utility Menu Options and Corresponding CLI Commands for Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Setup Utility Menu Option</th>
<th>Content Engine CLI Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT proxy caching</td>
<td>wmt license-key</td>
</tr>
<tr>
<td></td>
<td>wmt evaluate</td>
</tr>
<tr>
<td></td>
<td>wmt accept-license-agreement</td>
</tr>
<tr>
<td></td>
<td>wmt enable</td>
</tr>
<tr>
<td>WMT transparent caching</td>
<td>wccp router list</td>
</tr>
<tr>
<td></td>
<td>wccp wmt router-list</td>
</tr>
<tr>
<td></td>
<td>wccp version 2</td>
</tr>
<tr>
<td></td>
<td>wmt license-key</td>
</tr>
<tr>
<td></td>
<td>wmt evaluate</td>
</tr>
<tr>
<td></td>
<td>wmt accept-license-agreement</td>
</tr>
<tr>
<td></td>
<td>wmt enable</td>
</tr>
<tr>
<td>RealMedia proxy caching</td>
<td>rtsp proxy media-real license-key</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real evaluate</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real accept-license-agreement</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real enable</td>
</tr>
<tr>
<td>RealMedia transparent caching</td>
<td>wccp router list</td>
</tr>
<tr>
<td></td>
<td>wccp rtsp router-list</td>
</tr>
<tr>
<td></td>
<td>wccp version 2</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real license-key</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real evaluate</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real accept-license-agreement</td>
</tr>
<tr>
<td></td>
<td>rtsp proxy media-real enable</td>
</tr>
</tbody>
</table>

### Setup Utility Arrows and Keys

Table 4-4 describes the keys and arrows that you can use with the Setup utility.

### Table 4-4  Keys and Arrows for the Setup Utility

<table>
<thead>
<tr>
<th>Keys</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Displays information about why a particular menu item (for example, Network Configurations) is currently listed as “Incomplete.”</td>
</tr>
<tr>
<td>ESC</td>
<td>Quits the current dialog session and to return to the previous menu.</td>
</tr>
</tbody>
</table>
Before beginning the initial configuration of a Content Engine as a standalone device, you should decide which addressing scheme will be used for this Content Engine. The two supported addressing schemes for standalone Content Engines that are running the ACNS 5.x software are mutually exclusive:

- Manually specify a static IP address and network mask.
- Dynamically assign an IP address using the interface-level DHCP addressing scheme.

Note

Autoregistration needs to be disabled on the Content Engine before you can configure a Content Engine interface with interface-level DHCP or a static IP address. For information about how to disable autoregistration through the Setup utility, see Step 1 in the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.
If you do not enable interface-level DHCP on the Content Engine, you must manually specify a static IP address and network mask for the Content Engine. If the Content Engine moves to another location in another part of the network, you must manually enter a new static IP address and network mask for this Content Engine.

If you want to enable DHCP and are using the Setup utility to configure a standalone Content Engine, answer y when prompted (as described in Step 7 in the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21).

Do you want to enable DHCP on this interface? (y/n) [y]: y

If you are using the CLI method (instead of the Setup utility) to configure a standalone Content Engine, use the `ip address dhcp` interface configuration to enable interface-level DHCP on a standalone Content Engine.

### About Device Network Settings for Standalone Content Engines

In order to deploy a device as a standalone Content Engine on your network, you must initially configure a set of network settings on the Content Engine. These settings are collectively referred to as **device network settings**. After the device network settings are defined for the standalone Content Engine, it can become active on your network.

The device network settings that you should have before you start the basic configuration include the following:

- Host name of the Content Engine (for example, if you assign the Content Engine the name CE7305, the prompt will appear as:

  ```
  CE7305(config) #
  ```

- Internet Protocol (IP) domain name (for example, cisco.com)
- Administrator password
- IP addresses for the Content Engine

  If a static IP address is assigned to this Content Engine (for example, 10.0.1.2 as shown in Figure 4-4,) then you must also assign it an IP address network mask (for example, 255.255.255.0).

  **Tip** You also have the option of using interface-level DHCP to dynamically assign an IP address to a Content Engine interface instead of manually assigning a static IP address and network mask.

- Default gateway (for example, the router with address 10.0.1.1 is the default gateway for the Content Engine shown in Figure 4-4)
- DNS name server (for example, in Figure 4-4, the Content Engine will use the DNS server with the address 172.16.0.2 for domain name resolution)
Throughout the rest of this chapter, the term *general settings* is used to refer collectively to device network settings and disk configuration.

The ACNS 5.x software provides a Common Interface File System (CIFS) client and a Network File System (NFS) client for Content Engines to communicate with network attached storage (NAS) devices. For more information on this topic, see the “Mounting to a Network Attached Storage Device” section on page 5-13.

**Using the CLI Command Method to Configure General Settings for Standalone Content Engines**

To use the CLI command method (instead of the Setup utility) to configure the general settings on a standalone Content Engine, follow these steps:

**Step 1**
Open a console connection on the Content Engine, and log in to the Content Engine CLI using an ACNS system account that has superuser privileges. For more information see the “Using Telnet or a Console Session to Log in to a Standalone Content Engine” section on page 4-50.

**Step 2**
From privileged EXEC mode, enter global configuration mode to specify the general settings for this standalone Content Engine:

```
CE# config
```
Step 3 Configure the Ethernet interface on this Content Engine. You must do one of the following:

- To assign a static IP address and network mask (and not enable DHCP on this interface), enter the following command:

  ```
  CE(config)# interface {FastEthernet | GigabitEthernet} slot/port
  ip address ip-address netmask
  ```

- To enable interface-level DHCP, enter the following command:

  ```
  CE(config)# interface {FastEthernet | GigabitEthernet}
  slot/port ip address dhcp
  ```

If you configure your Ethernet interface using interface-level DHCP, then the remainder of the device network settings for this standalone Content Engine are automatically configured and you are finished with the configuration of the device network settings. If you manually assigned a static IP address, use the `ip default-gateway`, `ip name-server`, `hostname`, `ip domain-name`, and `primary-interface` global configuration commands to specify the remaining device network settings. Use the `disk config sysfs` global configuration command to use the Content Engine CLI command method to configure disk space.

---

**Launching the Setup Utility**

The Setup utility can be launched in these ways:

- Manually at any time by entering the `setup` privileged EXEC command at the CLI prompt
- Automatically when you initially boot up a device

---

**Note**

A Content Engine that is running the ACNS software comes with a single predefined superuser user account (root administrator). This predefined account can be used to invoke the Setup utility. The username for this predefined superuser user account is admin and the default password is default. If these defaults have been changed by another ACNS system administrator, you must obtain the new username and password.

---

**Manually Launching the Setup Utility**

To launch the Setup utility manually on a standalone Content Engine that already has its device network settings defined, follow these steps:

---

**Step 1** Using a login account that has the superuser privilege rights (privilege level of 15), log in to the Content Engine CLI through Telnet or Secure Shell (SSH) Version 1 or Version 2.

---

**Note**

For more information about logging in to the Content Engine CLI, see the “Using Telnet or a Console Session to Log in to a Standalone Content Engine” section on page 4-50. For more information about the different CLI modes, see the “ACNS Software CLI Command Modes for Standalone Content Engines” section on page B-8.
Step 2 Launch the Setup utility manually to configure one or more of the commonly used caching services (listed in Table 4-2).

ContentEngine# setup

The current basic configuration for this standalone Content Engine appears.

The displayed basic configuration also indicates which of the commonly used caching services are already configured on this Content Engine. In this case, only the HTTP proxy caching service is currently configured on this Content Engine.

Here is the current profile of this device:

<table>
<thead>
<tr>
<th>CDN device</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Proxy Caching</td>
<td>Yes</td>
</tr>
<tr>
<td>HTTP Transparent Caching</td>
<td>No</td>
</tr>
<tr>
<td>HTTP Reverse Proxy Caching</td>
<td>No</td>
</tr>
<tr>
<td>WMT Proxy Caching</td>
<td>No</td>
</tr>
<tr>
<td>WMT Transparent Caching</td>
<td>No</td>
</tr>
<tr>
<td>Real Media Proxy Caching</td>
<td>No</td>
</tr>
<tr>
<td>Real Media Transparent Caching</td>
<td>No</td>
</tr>
</tbody>
</table>

Note The displayed basic configuration indicates that this Content Engine is not a CDN device. This is because the standalone Content Engine is intentionally not registered with a Content Distribution Manager.

You can quickly modify the basic configuration of this standalone Content Engine by entering y when prompted as shown below.

Do you want to change this (y/n) [n]: y

For more information about how you can use the Setup utility to configure or modify the basic configuration of this Content Engine, see the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.

Automatically Launching the Setup Utility

When you initially boot up a device, follow these steps to launch the Setup utility automatically:

Step 1 Power up the Content Engine and open a console connection.

You must use a console connection rather than a Telnet session for initial configuration of these device network settings on the Content Engine. However, once you have used a console connection to define the device network settings, you can then use a Telnet session to perform subsequent configuration tasks on this Content Engine. For more information about using a console connection, see the “Using Telnet or a Console Session to Log in to a Standalone Content Engine” section on page 4-50.

After the operating system boots up, the following prompt appears:

ACNS boot:detected no saved system configuration

Do you want to enter basic configuration now?

hit RETURN to enter basic configuration:0019

At the appearance of this prompt, a 30-second countdown begins, during which you can launch the Setup utility.

Step 2 Press Enter.
Step 3  When prompted, enter the administrator password and press Enter.

admin password:

This is the case-sensitive password for the predefined superuser account. The password can include any printable character. By default, the username is admin and the password is default. The administrator password can be up to 20 characters long and is case sensitive. Each Content Engine in a farm must have a password. You must enter a password before pressing Enter.

Step 4  When prompted, reenter the administrator password and press Enter.

re-enter password:

The Setup utility is automatically launched, and you are prompted to specify the device mode for this particular device. When the Setup utility is launched on a Content Engine that supports device mode changes (for example, the CE-565), you are prompted to specify the device mode for that particular device.

Step 5  When prompted, press Enter or enter CE to specify the Content Engine device mode for this standalone Content Engine.

What is the mode of the device (CE/CR/CDM/PM) [CE]: CE

Step 6  When prompted, enter no to specify that this Content Engine is not going to be managed by a Content Distribution Manager.

Is this CE going to be managed by a CDM (Content Distribution Manager) (y/n) [y]: no

For more information about how to use the Setup utility after booting up a device for the first time, see the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.

Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine

To use the Setup utility to configure a basic configuration on a standalone Content Engine, follow these steps:

Step 1  Launch the Setup utility.

- If this is the first time that the device is being booted up, follow these steps:
  - Power up the device, open a console connection, and automatically invoke the Setup utility. (For detailed instructions, see the “Automatically Launching the Setup Utility” section on page 4-20.)
  - When the Setup utility is launched on a Content Engine that supports device mode changes (for example, the CE-565), you are prompted to specify the device mode for that particular device. When prompted, press Enter or enter CE to specify the Content Engine device mode for this standalone Content Engine.

What is the mode of the device (CE/CR/CDM/PM) [CE]: CE
When prompted, enter n to specify that this Content Engine is not going to be managed by a Content Distribution Manager.

Is this CE going to be managed by a CDM (Content Distribution Manager) (y/n) [y]: n

By default, the autoregistration is enabled on a Content Engine. When autoregistration is enabled on a Content Engine, the Content Engine automatically searches for and register with the Content Distribution Manager on the network. Because you want to deploy your Content Engine as a standalone device that is not registered with a Content Distribution Manager, you must specify n to disable autoregistration on this Content Engine.

Note: If you are using the CLI method to configure a standalone Content Engine (instead of the Setup utility), you can manually disable autoregistration by specifying the no auto-register enable global configuration command.

• Proceed to Step 2 below.

If the standalone Content Engine is running the ACNS 5.4 software and already has some of its basic configuration settings configured, follow these steps:

• Manually invoke the Setup utility, as described in the “Manually Launching the Setup Utility” section on page 4-19.

• Proceed to Step 2 below.

Step 2: When prompted, press Enter or enter y to indicate that you want to configure one or more of the seven commonly used caching services on this standalone Content Engine.

Do you want to configure this CE for doing HTTP Proxy Caching (y/n) [y]: y

Do you want to configure this CE for doing HTTP Transparent Caching using WCCP (y/n) [y]: y

Do you want to configure this CE for doing HTTP Reverse Proxy Caching using WCCP (y/n) [y]: y

Do you want to configure this CE for doing WMT Proxy Caching (y/n) [y]: y

Do you want to configure this CE for doing WMT Transparent Caching using WCCP (y/n) [y]: y

Do you want to configure this CE for doing Real Media Proxy Caching (y/n) [y]: y

Do you want to configure this CE for doing Real Media Transparent Caching using WCCP (y/n) [y]: y
The main menu of the Setup utility for a standalone Content Engine appears with the General Settings menu option highlighted. In the following example, the General Settings and Caching Related Configurations are currently reported as “Incomplete.”

```
+-----------------------------------------------------+
¦                      Main Menu                      ¦
+-----------------------------------------------------¦
¦->  General Settings               :Incomplete       ¦
¦    Caching Related Configurations :Incomplete       ¦
¦    Print Configuration                              ¦
¦    Exit (e)                                         ¦
+-----------------------------------------------------+
```

Main Menu

Configure network settings, disks, etc
Press '?' to see why this item is incomplete

**Note**
Even if you manually launch the Setup utility on this standalone Content Engine because the Content Engine already had some settings configured (for example, the device network settings, disk configuration, and HTTP proxy caching), the status of these configurations is listed as “Incomplete.” You must first use the Setup utility to accept the default (currently configured) values before the setting will be reported as “Complete” in a Setup menu. (When you are prompted for a value, any currently configured value is displayed as the default value. You can quickly accept the default value by pressing **Enter**.)

**Step 3**
If the general settings (device network settings and disk configuration) are already configured on this Content Engine, go to **Step 10** to configure one or more of the commonly used caching services on this Content Engine. Otherwise, complete **Step 4** through **Step 9** to configure the general settings for this standalone Content Engine.

**Step 4**
In the main menu, press **Enter** to choose the highlighted General Settings menu option. The General Settings submenu appears with the Network Configurations option highlighted.

```
+-----------------------------+
¦                          ¦
¦->  General Settings       ¦
¦| Disk Configurations      ¦Incomplete               ¦
¦| Print Configuration      ¦                           ¦
¦| Previous Menu (p)        ¦                           ¦
¦| Main Menu (m)            ¦                           ¦
¦| Exit (e)                 ¦                           ¦
+-----------------------------+
```

Main Menu

--- General Settings
Configure IP address, default gateway, name servers, domain name, etc
Press '?' to see why this item is incomplete
Step 5  Press Enter to choose the highlighted Network Configurations menu option.

The following warning appears.

WARNING: Changing any of the network settings from a telnet session may render the device inaccessible on the network. Therefore it is suggested that you have access to the console before modifying the network settings.

Step 6  After the warning appears, you are prompted to choose an interface identifier for the initial configuration of this Content Engine. Enter an interface identifier (for example, enter 1 to specify the Gigabit Ethernet 1/0 interface).

Please choose an interface to configure from the following list:
1: GigabitEthernet 1/0
2: GigabitEthernet 2/0

Enter choice: 1
Press the ESC key at any time to quit this session

Note  You can configure additional interfaces for this Content Engine through CLI commands at a later time, as described in the “Configuring Additional Network Interfaces” section on page 16-2.

Step 7  After specifying an interface identifier, you are asked if you want to enable interface-level DHCP on this particular interface.

- To not enable DHCP on this interface, go to Step 8.
- To enable DHCP on this interface, when prompted, enter y:

Do you want to enable DHCP on this interface? (y/n) [y]: y

Based on the input, the following CLIs will be configured:

interface GigabitEthernet 1/0
ip address dhcp
exit

Do you accept these configs (y/n) [y]: y

When you enter it again to accept the configuration, the remainder of your device network settings are automatically configured for this Content Engine. The main menu of the Setup utility appears and indicates that the configuration of the device network settings (Network Configuration) for this Content Engine is now “Complete.” Go to Step 9.

Step 8  Alternatively, to continue the configuration process using the static IP address method (and not enable interface-level DHCP on this Content Engine), follow these steps:

a.  When prompted, enter n:

Do you want to enable DHCP on this interface? (y/n) [y]: n

b.  When prompted for a local IP address, specify a static IP address (for example, 10.0.1.2).

Please enter the IP address of this interface: 10.0.1.2

c.  When prompted, specify the network mask (for example, 255.255.255.0)

Please enter the netmask of this interface: 255.255.255.0

d.  When prompted, specify the IP address of the gateway (for example, 10.0.1.1).

Please enter the default gateway: 10.0.1.1
e. When prompted, specify the IP address of the DNS server (for example, 172.16.0.2).
   Please enter the domain name server IP: 172.16.0.2

f. When prompted, specify the IP domain name of this Content Engine (for example, cisco.com).
   Please enter the domain name: cisco.com

g. When prompted, specify the host name of this Content Engine (for example, CE7305).
   Please enter the hostname: CE7305

A message appears, indicating which CLI commands will be configured based on your input.

Based on the input, the following CLIs will be configured:

   interface GigabitEthernet 1/0
       ip address 10.0.1.2 255.255.255.0
       exit
       ip default-gateway 10.0.1.1
       ip name-server 172.16.0.2
       ip domain-name cisco.com
       hostname CE7305

h. When prompted, enter y to save these network configurations on this standalone Content Engine.

Do you accept these configs (y/n) [y]: y

The main menu appears, indicating that the configuration of the device network settings (network configuration) for this standalone Content Engine is now complete.

---

**Note**

You are now finished with the configuration of the device network settings for this Content Engine, which is now standalone. The next step is to configure the disk space on this standalone Content Engine. Go to Step 9.

---

**Step 9**

Configure the disk configuration for this standalone Content Engine, as follows:

a. From the main menu, choose the General Settings option. In the displayed General Settings submenu, choose the Disk Configurations option. The current storage allocation for this standalone Content Engine appears as shown in the following example:

Here is the current storage allocation scheme:

<table>
<thead>
<tr>
<th>File System</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSFS</td>
<td>29.9GB</td>
</tr>
<tr>
<td>CFS</td>
<td>0.0GB</td>
</tr>
<tr>
<td>MEDIAFS</td>
<td>0.0GB</td>
</tr>
<tr>
<td>CDNFS</td>
<td>1.0GB</td>
</tr>
</tbody>
</table>

Do you want to change this (y/n) [n]:

Disk space in ACNS software is allocated on a per-file system basis, rather than on a per-disk basis. You can configure your overall disk storage allocations according to the kinds of client protocols you expect to use and the amount of storage that you need to provide for each of the functions, as described in this table.
b. When prompted, enter y to change the current storage allocations on this Content Engine.

Do you want to change this (y/n) [n]: y

The following questions will prompt you how the available storage is to be allocated to different file systems. You can either enter an absolute amount of storage in GB or MB, or a percentage of the available storage. In the former case, the desired amount should be entered followed by either "GB" or "MB", and in the latter case, the number should be followed by "%"

c. When prompted, enter the amount of storage to be allocated to the system file system (sysfs). For example, enter 2GB.

Please enter the amount of storage to be allocated to SYSFS (This file system is used for storing user and logging files; at least 1GB required): 2GB

d. When prompted, enter the amount of storage to be allocated to the cache file system (cfs). For example, enter 20MB.

Please enter the amount of storage to be allocated to CFS (This file system is used for storing HTTP objects): 20MB

e. When prompted, enter 0GB or 0% to specify that no amount of storage is to be allocated to the ACNS network file system (cdnfs).

Please enter the amount of storage to CDNFS (This file system is used for prepositioned content): 0GB

Note: You do not want to allocate any storage to the cdnfs because this file system is used to store pre-positioned content on a registered Content Engine. You cannot pre-position content on a standalone Content Engine. However, you can preload content on a standalone Content Engine at a later time, as described in the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2.

f. When prompted, enter the amount of storage to be allocated to the media file system (mediafs). If you plan to enable WMT caching or RTSP caching on this standalone Content Engine, then you must allocate storage for the mediafs. For example, enter 10MB.

Please enter the amount of storage to be allocated to MEDIAFS (This file system is used for storing WMT and Real media content): 10MB

<table>
<thead>
<tr>
<th>Disk Storage Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysfs (system file system)</td>
<td>Stores log files, including transaction logs, syslogs, and internal debugging logs. Also can store image files and configuration files.</td>
</tr>
<tr>
<td>cfs (cache file system)</td>
<td>Caches HTTP and FTP objects.</td>
</tr>
<tr>
<td>mediafs (media file system)</td>
<td>Caches content that is fetched through the two streaming protocols (RTSP and WMT). By default 30 percent of the mediafs space is reserved for RTSP streaming content, and 70 percent is reserved for WMT streaming content.</td>
</tr>
</tbody>
</table>
The new disk configuration for this standalone Content Engine appears, as shown in the following example:

Here is the new disk configuration:

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSFS</td>
<td>2GB</td>
</tr>
<tr>
<td>CFS</td>
<td>20MB</td>
</tr>
<tr>
<td>CDNFS</td>
<td>0GB</td>
</tr>
<tr>
<td>MEDIAFS</td>
<td>10MB</td>
</tr>
</tbody>
</table>

When prompted, enter y to accept the new disk configuration.

Do you accept these configs (y/n) [y]: y

The General Settings submenu appears and the Disk Configurations option is now reported as “Complete.” The general settings (network configuration settings and the disk configurations) are now configured on this standalone Content Engine.

---

**Step 10** Configure one or more of the commonly used caching services (see Table 4-2 for list) on this standalone Content Engine, as follows:

**a.** Use one of the following methods to display the Caching Related Configurations submenu:

- From the General Settings submenu, choose the Main Menu option to return to the main menu. The Caching Related Configurations menu option is currently reported as “Incomplete” because you have not configured any caching services on this Content Engine yet.
- From the main menu, choose the **Caching Related Configurations** option to configure one or more commonly used caching services on this Content Engine. The Caching Related Configurations submenu appears with a list of the caching services that you previously selected. In the following example all of these caching services are listed as “Incomplete” because you have not configured any of them yet:

```
+--------------------------+
| Caching Related Configurations |
+--------------------------+
| -> HTTP Caching           :Incomplete |
| WMT Caching               :Incomplete |
| Real Media Caching        :Incomplete |
| Print Configuration       |
| Previous Menu (p)         |
| Main Menu (m)             |
| Exit (e)                  |
+--------------------------+
```

**Main Menu**

---» Caching Related Configurations

**Configure HTTP Caching**

b. To configure HTTP caching on this standalone Content Engine, complete **Step 11**.

c. To configure WMT caching on this standalone Content Engine, complete **Step 15**.

d. To configure RealMedia caching on this standalone Content Engine, complete **Step 18**.

**Step 11**

Configure HTTP caching as follows:

a. From the Caching Related Configurations submenu, choose the **HTTP Caching** option.

The HTTP Caching submenu appears with options for each of the available HTTP caching services. In the following example, all three of the HTTP caching services (HTTP proxy caching, HTTP transparent caching, and HTTP reverse proxy caching) are to be configured on this Content Engine:

```
+--------------------------+
| HTTP Caching             |
+--------------------------+
| -> HTTP Proxy Caching    :Incomplete |
| HTTP Transparent Caching :Incomplete |
| HTTP Reverse Proxy Caching :Incomplete |
| Print Configuration      |
| Previous Menu (p)        |
| Main Menu (m)            |
| Exit (e)                 |
+--------------------------+
```

**Main Menu**

---» Caching Related Configurations

---» HTTP Caching

**Configure this CE for doing HTTP Proxy caching**

b. To configure HTTP proxy caching, complete **Step 12**.

c. To configure HTTP transparent caching using WCCP, complete **Step 13**.

d. To configure HTTP reverse proxy caching, complete **Step 14**.
**Step 12** Configure HTTP proxy caching as follows:

a. From the HTTP Caching submenu, choose the **HTTP Proxy Caching** option.

b. When prompted, specify the incoming proxy ports for proxy-style HTTP requests from client browsers. These are the port numbers on which this standalone Content Engine will accept incoming proxy-style HTTP requests. These are also the ports that the Content Engine will use to serve the requested content to the requester (the client browser). For HTTP proxy caching, this standalone Content Engine is functioning as a nontransparent forward proxy server that receives HTTP requests directly from the client browsers.

The incoming proxy port numbers can be from 1 to 65535. You can specify up to eight incoming proxy ports, each separated by a space. The incoming proxy ports can be the same ports that are used by transparent mode services (for example, HTTP transparent caching) on this standalone Content Engine.

Please enter all the HTTP Proxy incoming ports (up to 8, separated by spaces) [80 8080]: 80 8080 8081

A list of the configured CLI commands appears and you are asked if you want save this configuration. Enter **y** to accept this configuration.

Based on the input, the following CLIs will be configured:

http proxy incoming 80 8080 8081

C. When prompted, enter **y** to accept this configuration.

Do you accept these configs (y/n) [y]: y

The specified settings are saved and the HTTP Caching submenu reappears. The HTTP Proxy Caching option is now listed as “Complete.” Remember that you still must configure the client browsers to point directly to this Content Engine as their HTTP proxy server, as described in the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.

**Step 13** Configure HTTP transparent caching using WCCP as follows:

a. From the HTTP Caching submenu, choose the **HTTP Transparent Caching** option.

b. When prompted, specify the IP addresses of the WCCP Version 2-enabled routers that will transparently redirect HTTP requests to this standalone Content Engine on port 80 only, or press **Enter** to use the default gateway. In this case, the default gateway (the WCCP Version 2-enabled router that was specified as this Content Engine’s default gateway) has an IP address of 10.0.1.1.

Please enter the IP addresses of WCCP routers [10.0.1.1]:

A list of the configured CLI commands appears, along with a reminder that you must configure the web-cache service (WCCP service 0) on the WCCP Version 2-enabled router.

NOTE: Please remember to configure web-cache service on the router.

Based on the input, the following CLIs will be configured:

wccp router-list 1 10.0.1.1
wccp version 2
wccp web-cache router-list 1
d. Enter y to accept this configuration.

Do you accept these configs (y/n) [y]: y

The specified settings are saved and the HTTP Caching submenu reappears. The HTTP Transparent Caching option is now listed as “Complete.” Remember that you still must configure the web-cache service (WCCP service 0) on the WCCP Version 2-enabled routers, as described in the “Configuring WCCP Services on a Router” section on page 6-27.

Step 14 Configure HTTP reverse proxy caching using WCCP as follows:

a. From the HTTP Caching submenu, choose the HTTP Reverse Proxy Caching option.

b. When prompted, specify the IP addresses of the WCCP Version 2-enabled routers that will redirect reverse proxy packets to this Content Engine, or press Enter to use the default gateway (for example, the WCCP Version 2-enabled router with the IP address of 10.0.1.1).

Please enter the IP addresses of WCCP routers [10.0.1.1]:

c. Enter y to enable HTTP reverse proxy caching on this Content Engine.

Do you want to enable HTTP reverse proxy caching (y/n) [y]: y

A list of the configured CLI commands appears, along with a reminder that you must still configure WCCP service 99 (reverse proxy caching) on the WCCP Version 2-enabled router.

NOTE: Please remember to configure service 99 on the router.

Based on the input, the following CLIs will be configured:

```
  wccp router-list 1 10.0.1.1
  wccp version 2
  wccp reverse-proxy router-list 1
```

d. Enter y to accept this configuration.

Do you accept these configs (y/n) [y]: y

The specified settings are saved and the HTTP Caching submenu reappears. The HTTP Reverse Proxy Caching option is now listed as “Complete.” Remember to configure the reverse proxy caching service (WCCP service 99) on the WCCP Version 2-enabled router, as described in the “Configuring the Reverse-Proxy Service (Service 99) on a Router” section on page 6-33.

Step 15 Configure WMT caching as follows:

a. Use one of the following methods to display the WMT Caching submenu.

- From the HTTP Caching submenu, choose the Previous Menu option and then choose the WMT Caching option from the Caching Related Configurations submenu.
- From the Caching Related Configurations submenu, choose the WMT Caching option.
From the main menu, choose the **Caching Related Configurations** option and then choose the **WMT Caching** option from the Caching Related Configurations submenu.

In the following example, both WMT caching services (WMT proxy caching and WMT transparent caching) are to be configured on this standalone Content Engine:

```
+----------------------------------------------+
¦                 WMT Caching                  ¦
+----------------------------------------------¦
¦->  WMT Proxy Caching       :Incomplete     ¦
¦    WMT Transparent Caching :Incomplete     ¦
¦    Print Configuration                       ¦
¦    Previous Menu (p)                         ¦
¦    Main Menu (m)                             ¦
¦    Exit (e)                                  ¦
+----------------------------------------------+
```

**Main Menu**
```
---> Caching Related Configurations
---> WMT Caching
```

**Configure this CE for doing WMT Proxy caching**

b. To configure WMT proxy caching, complete [Step 16](#).

c. To configure WMT transparent caching using WCCP Version 2, complete [Step 17](#).

**Step 16** Configure WMT proxy caching, as follows:

a. From the WMT caching submenu, choose the **WMT Proxy Caching** option.

b. If there is a WMT license already installed on this Content Engine, enter **n** when the following prompt appears:

   WMT license key is already installed. Do you want to install a different license key (y/n) [n]: **n**

   Otherwise, specify whether or not you have a license key for WMT when prompted. If you have your Cisco license key for the WMT product, then enter **y**. Otherwise, enter **n** to use the evaluation license for the WMT feature on this standalone Content Engine (as shown in this example).

   Do you have the license key for WMT (y/n) [y]: **n**

   Do you want to evaluate WMT (y/n) [y]: **y**

**Note** When you use the Setup utility to configure WMT proxy caching, the Content Engine automatically is configured to use the default port (port 1755) to listen for incoming WMT requests.

A list of the configured CLI commands appears, and you are asked if you want save this configuration. If you used the evaluation license to enable WMT on this Content Engine, the `wmt evaluate` command is included in the list of constructed CLI commands. If you used your Cisco WMT license, then the `wmt license-key` command is listed instead of the `wmt evaluate` command.

Based on the input, the following CLIs will be configured:

```
wmt evaluate
wmt accept-license-agreement
wmt enable
```
c. When prompted, enter \texttt{y} to accept this configuration.

\texttt{Do you accept these configs (y/n)} [\texttt{y}]: \texttt{y}

The specified settings are saved and the WMT Caching submenu reappears. The WMT Proxy Caching option is now listed as “Complete.” Remember that you still must configure Windows Media Player on the end user desktops to point directly to this Content Engine as their proxy server.

\textbf{Tip} You can also use the Content Engine CLI to configure WMT streaming on this Content Engine at a later time, as described in \textit{Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”}

\textbf{Step 17} Configure WMT transparent caching as follows:

\textbf{a.} From the WMT caching menu, choose the \textbf{WMT Transparent Caching} option.

\textbf{b.} When prompted, specify the IP addresses of the WCCP Version 2-enabled routers that will redirect WMT requests to this Content Engine. Press \texttt{Enter} to use the default gateway, or enter the IP addresses of other WCCP Version 2-enabled routers that you want to redirect WMT requests to this Content Engine. In this case, the default gateway (the WCCP Version 2-enabled router that was specified as this Content Engine’s default gateway) has an IP address of 10.0.1.1.

\texttt{Please enter the IP addresses of WCCP routers [10.0.1.1]:}

c. If there is a WMT license already installed on this Content Engine, enter \texttt{n} when the following prompt appears:

\texttt{WMT license key is already installed. Do you want to install a different license key (y/n)} [\texttt{n}]: \texttt{n}

Otherwise, specify whether or not you have a license key for WMT when prompted. If you have your Cisco license key for the WMT product, then enter \texttt{y}. Otherwise, enter \texttt{n} to use the evaluation license for the WMT feature on this standalone Content Engine (as shown in this example).

\texttt{Do you have the license key for WMT (y/n)} [\texttt{y}]: \texttt{n}

\texttt{Do you want to evaluate WMT (y/n)} [\texttt{y}]: \texttt{y}

d. When prompted, enter \texttt{y} to enable WMT transparent caching on this Content Engine.

\texttt{Do you want to enable WMT transparent caching (y/n)} [\texttt{y}]: \texttt{y}

A list of the configured CLI commands appears. If you used the evaluation license to enable WMT on this Content Engine, the \texttt{wmt evaluate} command is included in the list of configured CLI commands. If you used your Cisco WMT license, then the \texttt{wmt license-key} command is listed instead of the \texttt{wmt evaluate} command.

Based on the input, the following CLIs will be configured:

\begin{verbatim}
  wmt evaluate
  wmt accept-license-agreement
  wmt enable
  wccp router-list 1 10.0.1.1
  wccp version 2
  wccp wmt router-list 1
\end{verbatim}

e. When prompted, enter \texttt{y} to accept this configuration.

\texttt{Do you accept these configs (y/n)} [\texttt{y}]: \texttt{y}

The specified settings are saved and the WMT Caching submenu appears. The WMT Transparent Caching option is now listed as “Complete.”
**Step 18** Configure RealMedia caching, as follows:

a. Use one of the following methods to display the RealMedia Caching submenu.
   - From the WMT Caching submenu, choose the Previous Menu option and then choose the WMT Caching option from the Caching Related Configurations submenu (shown below).
   - From the Caching Related Configurations submenu, choose the Real Media Caching option.
   - From the main menu, choose the Caching Related Configurations option and then choose the Real Media Caching option from the Caching Related Configurations submenu.

In the following example, both RealMedia caching services (RealMedia proxy caching and Real Media transparent caching using WCCP Version 2) are to be configured on this standalone Content Engine:

```
+-----------------------------------------------+
¦              Real Media Caching               ¦
+-----------------------------------------------¦
¦-> Caching       :Incomplete                  ¦
¦    Real Transparent Caching :Incomplete      ¦
¦    Print Configuration                        ¦
¦    Previous Menu (p)                          ¦
¦    Main Menu (m)                              ¦
¦    Exit (e)                                   ¦
+-----------------------------------------------+
```

b. To configure RealMedia proxy caching, complete **Step 19**.

c. To configure RealMedia transparent caching using WCCP Version 2, complete **Step 20**.

**Step 19** Configure RealMedia proxy caching, as follows:

a. From the RealMedia Caching submenu, choose the **Real Proxy Caching** option.

b. If there a RealProxy license already installed on this Content Engine, enter **n** when the following prompt appears:

```
Real Proxy license key is already installed. Do you want to install a different license key (y/n) [n]:n
```

Otherwise, specify whether or not you have a RealProxy license key. If you have your Cisco RealProxy license key, then enter **y**. Otherwise, enter **n** to use the RealProxy evaluation license for this standalone Content Engine (as shown in this example).

```
Do you have the license key for Real Proxy (y/n) [y/n] :n
Do you want to evaluate Real Proxy (y/n) [y]:y
```

A list of the configured CLI commands appears, and you are asked if you want save this configuration. If you used the evaluation license to enable the RTSP proxy for RealMedia requests, the **rtsp proxy media-real evaluate** command is included in the list of configured CLI commands.

Based on the input, the following CLIs will be configured:

```
rtsp proxy media-real accept-license-agreement
rtsp proxy media-real enable
rtsp proxy media-real evaluate
```
c. When prompted, enter y to accept this configuration.
   Do you accept these configs (y/n) [y]: y

   **Note** When you use the Setup utility to configure RealMedia proxy caching, the Content Engine automatically is configured to use the standard RTSP port (default port 554) to listen for incoming RealMedia requests. The RTSP gateway is the single point of entry for RTSP messages on the standalone Content Engine. The RTSP gateway runs on the Content Engine and is automatically enabled. By default, the RTSP gateway listens on port 554 for incoming RTSP requests. If you want to configure the RTSP gateway to listen for incoming RTSP requests on a port other than the default port (port 554), you must change the incoming RTSP port. You must use the Content Engine CLI (the `rtsp port incoming` command) to change the RTSP incoming port on a standalone Content Engine. For more information, see the “Configuring Basic Settings for the RTSP Gateway” section on page 8-16.

   The specified settings are saved and the RealMedia Caching submenu reappears. The RealProxy Caching option is now listed as “Complete.” Remember that you still must configure the RealMedia players on the client desktops to point directly to this Content Engine as their proxy server, as described in the “Pointing RealMedia Players Directly to a Standalone Content Engine” section on page 4-46.

   **Tip** You can also configure RealMedia streaming (VOD files and live splitting) on this Content Engine at a later time, as described in Chapter 8, “Configuring RealMedia Services on Standalone Content Engines.”

---

**Step 20** Configure RealMedia transparent caching using WCCP as follows:

a. From the RealMedia Caching submenu, choose the **Real Transparent Caching** option.

b. When prompted, specify the IP addresses of the WCCP Version 2-enabled routers that will redirect RealMedia requests to this Content Engine. Press Enter to use the default gateway, or enter the IP addresses of other WCCP Version 2-enabled routers that you want to redirect RealMedia requests to this Content Engine. In this case, the default gateway has an IP address of 10.0.1.1 (the WCCP Version 2-enabled router that was specified as this Content Engine’s default gateway).

   Please enter the IP addresses of WCCP routers [10.0.1.1]:

   c. If there is a RealProxy license already installed on this Content Engine, enter n when the following prompt appears:

   ```
   Real Proxy license key is already installed. Do you want to install a different license key (y/n) [n]: n
   ```

   Otherwise, specify whether or not you have a RealProxy license key. If you have your Cisco RealProxy license key, then enter y. Otherwise, enter n to use the RealProxy evaluation license for this standalone Content Engine (as shown here in this example).

   ```
   Do you have the license key for Real Proxy (y/n) [y]: n
   Do you want to evaluate Real Proxy (y/n) [y]: y
   ```
d. Enter y to enable RealMedia transparent caching on this Content Engine.

Do you want to enable Real Media transparent caching \( y/n \) \( \{y\} \):

A list of the configured CLI commands appears along with a reminder that you still must configure the rtsp service (service 80) on the WCCP Version 2-enabled router. If you used the evaluation license to enable the RTSP proxy for RealMedia requests, the rtsp proxy media-real evaluate command is included in the list of configured CLI commands.

NOTE: Please remember to configure service 80 on the router.

Based on the input, the following CLIs will be configured:
- rtsp proxy media-real accept-license-agreement
- rtsp proxy media-real enable
- wccp router-list 1 10.0.1.1
- wccp version 2
- wccp rtsp router-list 1

e. When prompted, enter y to accept this configuration.

Do you accept these configs \( y/n \) \( \{y\} \):

The specified settings are saved and the RealMedia Caching submenu reappears. The Real Transparent Caching option is now listed as “Complete.” Remember that you still must configure the rtsp service (service 80) on the WCCP Version 2-enabled router, as described in the “Configuring the RTSP Service (Service 80) on a Router” section on page 6-30.

Note: If the Content Engine is behind a network address translation (NAT)-enabled router, you must also specify the IP address of the RTSP gateway. After you have used the Setup utility to complete the basic configuration of a Content Engine, you can exit the Setup utility and then use the Content Engine CLI to specify the IP address of the RTSP gateway. To specify the IP address of the RTSP gateway, use the rtsp ip-address rtsp-gateway-ip-address global configuration command, as described in the “Configuring Basic Settings for the RTSP Gateway” section on page 8-16.

### Configuring Client Browsers and Media Players for Direct Proxy Routing

Remember that after you configure the standalone Content Engine for nontransparent (proxy) caching, you must configure the client browsers and media players to route their content requests directly to this Content Engine (direct proxy routing). For information about how to point client browsers or media players to a standalone Content Engine that is functioning as a nontransparent proxy server for these clients, see Table 4-5.
Table 4-5  Configuring Client Browsers and Media Players to Support Direct Proxy Routing of Content Requests

<table>
<thead>
<tr>
<th>Nontransparent Caching</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP proxy caching</td>
<td>See the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.</td>
</tr>
<tr>
<td>HTTPS proxy caching</td>
<td>See the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.</td>
</tr>
<tr>
<td>WMT proxy caching</td>
<td>See the See the “Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests” section on page 4-43.</td>
</tr>
<tr>
<td>RealMedia proxy caching</td>
<td>See the “Pointing RealMedia Players Directly to a Standalone Content Engine” section on page 4-46.</td>
</tr>
</tbody>
</table>

**Pointing Client Browsers Directly to a Standalone Content Engine**

If nontransparent caching is to be used to direct content requests to a standalone Content Engine that is functioning as a nontransparent proxy server, you must configure the client browsers to point directly to this Content Engine. To point a client browser to a Content Engine you can use proxy autoconfiguration (PAC) files, or you can manually configure the browser to point to a specific standalone Content Engine.

For more information on these two different methods, see the following sections:

- Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine, page 4-37
- Manually Pointing Client Browsers to a Standalone Content Engine, page 4-42
Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine

The ACNS 5.x software provides support for PAC files to point client browsers directly to a standalone Content Engine (nontransparent forward proxy server for these client browsers). A PAC file is a configuration file that is written in JavaScript and stored on an FTP server in your intranet. To use PAC files to point client browsers directly to a standalone Content Engine, follow these steps:

**Step 1** Create the PAC file on an FTP server.

**Step 2** Download the PAC file from the FTP server to the Content Engine that will act as the proxy server for the client browsers.

Each time you download a new PAC file to a standalone Content Engine, follow these steps:

- **a.** Disable proxy autoconfiguration on the Content Engine (**no proxy-auto-config enable** command).
- **b.** Download the new PAC file to the Content Engine.

**Note** You must configure disks /local1 or /local2 as a sysfs volume on the Content Engine before downloading the autoconfiguration file to either of these two disk locations.

- **c.** Enter the **proxy-auto-config enable** command to reenable the automatic proxy configuration feature on the Content Engine.

Manually configure the browser for automatic proxy configuration by explicitly specifying the Content Engine’s IP address, incoming port number, file directory, and the name of the PAC file in the browser.

**Note** Microsoft Internet Explorer and Netscape browsers support the use of PAC files.

**Step 3** If a browser is configured for automatic proxy configuration, then when the browser starts up, it will obtain the necessary proxy information (for example, the proxy server’s IP address and port configuration information) from the PAC file (.pac file).

The following is an example of how to use a PAC file to point client browsers to the Content Engine:

**Step 1** Create a PAC file on an FTP server.

The following is an example of a very simple PAC file named proxyfile.pac. In this case, there is only one proxy (one standalone Content Engine) and traffic that is not destined for “cisco.com” is sent to the proxy server (Content Engine) for all Internet requests.

Example #1: Use proxy for everything except local hosts
This would work in Netscape’s environment. All hosts which aren’t fully qualified, or the ones that are in local domain, will be connected to directly. Everything else will go through w3proxy:8080. If the proxy goes down, connections become automatically direct.

```javascript
function FindProxyForURL(url, host) {
    if (isPlainHostName(host) ||
        dnsDomainIs(host, ".cisco.com")
    )
        return "DIRECT";
    else
        return "PROXY ce1.cisco.com:8080; DIRECT";
```
Note: This is the simplest and most efficient autoconfig file for cases where there's only one proxy.

Step 2  Download the PAC file from the specified FTP server to the Content Engine. By default, the PAC file is downloaded to the present working directory of the Content Engine.

This example shows how to download a PAC file named the proxyfile.pac from an FTP server that has an IP address of 172.16.10.10 to the Content Engine. The Content Engine is functioning as a PAC file server because the client browsers will be pointed to this PAC file when the browsers are started up.

```
ContentEngine# proxy-auto-config download 172.16.10.10 remotedirname proxyfile.pac
```

Step 3  Enable the browser autoconfiguration feature on this standalone Content Engine.

```
ContentEngine(config)# proxy-auto-config enable
```

Step 4  Manually configure each client browser for automatic proxy configuration. If a browser is configured for automatic proxy configuration, the browser will obtain the necessary information from the specified .pac file on the Content Engine each time the browser starts up. You must explicitly specify the Content Engine’s IP address, incoming port number, file directory, and name of the .pac file in the browser.

The following steps show how to perform this task from Internet Explorer Version 6.0:

a. From the Internet Explorer GUI, choose Tools > Internet Options. The Internet Options window appears. (See Figure 4-5.)

![Figure 4-5 Internet Options Window](image)

b. At the top of the Internet Options window, click the Connections tab to bring this tab to the front. (See Figure 4-6.)
c. On the Connections tab, click the Settings button. The Settings window appears. (See Figure 4-7.)

![Connections Tab Window](image1.png)

**Figure 4-6  Connections Tab Window**

![Settings Window](image2.png)

**Figure 4-7  Settings Window**

d. Check the Use automatic configuration script check box.

e. In the Address field, enter the URL of the .pac file that this browser should use to determine which proxy server (the Content Engine) it should direct its content requests to.

```
http://ContentEngine-IPaddress:portnumber/pac filename
```
In the following example, the URL of the .pac file specifies that the .pac file is named proxyfile.pac, and is stored on a Content Engine (nontransparent forward proxy server) that has an IP address of 172.16.10.10 and 8080 as an incoming port of 8080.

http://172.16.10.10:8080/proxyfile.pac

Note When specifying the port number in the URL of the .pac file, use the same port number that was specified as the proxy's incoming port number (through the `http proxy incoming portnumber` global configuration command, or the Setup utility as described in Step 12 of “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine”). For instance, if port 8080 is specified with the `http proxy incoming 8080` command, then use 8080 as your port number in the URL of the .pac file.

f. Click OK to save the settings and close the Settings window.

Step 5 If NTLM is used in this environment to control user Internet access, you must change the default setting for user authentication in the client browser to prevent the user from being prompted by a popup window to log in every time the user attempts to access a new website. The following steps show how to change this default setting in Internet Explorer Version 6.0:

a. In the Internet Options window (Figure 4-5), click the Security tab. The Security tab appears with the Internet selected as the web content zone. (See Figure 4-8).

b. Click the Custom Level button. The Security Settings window appears.

c. In the Security Settings window, scroll down to the User Authentication section of the window. (See Figure 4-9.)
Figure 4-9 shows that the default user authentication setting is for automatic logon for an intranet zone only.

d. Change the default user authentication setting by clicking the **Automatic logon with current username and password** radio button. (See Figure 4-10).

e. Click **OK** to close the Security Settings window.
Manually Pointing Client Browsers to a Standalone Content Engine

To manually point a client browser to a standalone Content Engine instead of using proxy autoconfiguration (PAC) files, you must explicitly specify the IP address and port number of the Content Engine (nontransparent proxy server for this client browser) in the browser.

The following example describes how to perform this task from Internet Explorer, Version 6.0:

**Step 1** From the Internet Explorer GUI, choose **Tools > Internet Options**. The Internet Options window appears. (See Figure 4-5.)

**Step 2** At the top of the Internet Options window, click the Connections tab to bring it to the front. (See Figure 4-6.)

**Step 3** On the Connections tab, click the **Settings** button. The Settings window appears. (See Figure 4-7.)

**Step 4** In the Settings window, check the **Use a proxy server for this connection** check box to manually point the browser directly to a Content Engine. (See Figure 4-11.)

**Figure 4-11 Manually Pointing a Browser Directly to a Content Engine**

**Step 5** In the Address field, enter the IP address of the Content Engine that you want this browser to point to. Specify the IP address of the Content Engine that this client browser will directly send its content requests. For example, to specify the Content Engine that has an IP address of 172.16.10.10 as the direct proxy server, enter **172.16.10.10** into the Address field.
Configuring Client Browsers and Media Players for Direct Proxy Routing

Step 6 In the Port field, enter the port number of the standalone Content Engine that will be the proxy server for this browser.

**Note** Use one of the port numbers that you specified as an incoming proxy port when you configured HTTP proxy caching on the standalone Content Engine (through the Setup utility as described in Step 12 of “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine,” or by entering the `http proxy incoming portnumber` global configuration command). For instance, if port 8080 is specified as the incoming proxy port (`http proxy incoming 8080` command), then enter 8080 as your port number in the Port field.

Step 7 Click OK.

Step 8 If NTLM is used in this environment to control user Internet access, you must change the default setting for user authentication in the client browser to prevent the user from being prompted by a popup window to log in every time the user attempts to access a new website.

For an example of how to change this default setting in Internet Explorer Version 6.0, see Step 5 on page 4-40.

---

**Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests**

If direct proxy routing is being used instead of WMT transparent redirection to direct Windows Media 9 players requests to send their WMT RTSP requests directly to a standalone Content Engine, you must configure Windows Media 9 players on client desktops to point directly to the Content Engine (a nontransparent WMT proxy server for these web clients).

The Content Engine must be running the ACNS 5.3.1 software and later releases to support direct proxy routing of WMT RTSP requests from Windows Media 9 players.

To explicitly configure Windows Media 9 players on client desktops to point directly to a specific standalone Content Engine, follow these steps:

**Step 1** Open Windows Media player on the client desktops.

**Step 2** In the Windows Media player menu bar, choose **Tools > Options**. The Options window appears.

**Step 3** In the Options window, click the Network tab. (See Figure 4-12.)
Step 4  On the Network tab under Streaming protocols, click the Multicast, UDP, TCP and HTTP check boxes if they are not already selected.

Step 5  On the Network tab, under Streaming proxy settings, choose RTSP and click Configure. The Configure Protocol window appears. (See Figure 4-13.)

**Figure 4-13  Configuring RTSP as a Streaming Protocol**

Step 6  Click the Use the following proxy server radio button.

Step 7  In the Address field, enter the IP address of the Content Engine (the nontransparent WMT proxy server for RTSP requests from this Windows Media 9 player).
Step 8  In the Port field, enter the port number on which the Content Engine will accept incoming WMT RTSP requests from this Windows Media 9 player. Port 554 is the default port for RTSP incoming requests. By default, the Content Engine listens on port 554 for incoming RTSP requests. If you have changed this default port setting on the Content Engine (that is, you have used the `rtsp port incoming port-number` global configuration command to configure the Content Engine to listen for incoming RTSP requests on a port other than port 554), make sure that you enter that port number into the Port field of the Configure Protocol window.

Step 9  Click OK to close the Configuration Protocol window.

Step 10  Click Apply to apply the settings to the Windows Media 9 player.

Step 11  Click OK to close the Options window in the Windows Media 9 player.

---

Pointing Windows Media Players Directly to a Standalone Content Engine for WMT MMS Requests

**Note**  In ACNS 5.5 software release, MMS requests are transparently redirected to the HTTP or RTSP protocol, and the request is served over RTSP or HTTP in the same order.

If direct proxy routing is being used to direct WMT MMS requests from Windows Media players directly to a standalone Content Engine, you must configure Windows Media player on client desktops to point directly to the Content Engine (a nontransparent WMT proxy server for these web clients).

To explicitly configure Windows Media players on client desktops to send their WMT MMS requests directly to a specific standalone Content Engine, follow these steps:

---

Step 1  Open Windows Media player on the client desktops.

Step 2  In the Windows Media player menu bar, choose Tools > Options. The Options window appears.

Step 3  In the Options window, click the Network tab. (See Figure 4-12.)

Step 4  On the Network tab under Streaming protocols, click the Multicast, UDP, TCP and HTTP check boxes if they are not already selected.

Step 5  On the Network tab, under Streaming proxy settings, choose MMS, and click Configure. The Configure Protocol window appears. (See Figure 4-14.)
Step 6  Click the **Use the following proxy server** radio button.

Step 7  In the Address field, enter the IP address of the Content Engine (the nontransparent WMT proxy server for this Windows Media player).

Step 8  In the Port field, enter the port number on which the Content Engine will accept incoming WMT MMS requests from this Windows Media player.

By default, the Content Engine listens on port 1755 for incoming MMS requests from WMT clients. When you use the Setup utility to configure WMT proxy caching, the Content Engine automatically is configured to use the default port (port 1755) to accept incoming MMS requests.

Step 9  Click **OK** to close the Configuration Protocol window.

Step 10  Click **Apply** to apply the settings to the Windows Media player.

Step 11  Click **OK** to close the Options window in the Windows Media player.

---

### Pointing RealMedia Players Directly to a Standalone Content Engine

If direct proxy routing is being used instead of RTSP transparent redirection to direct RTSP requests from RealMedia players directly to a standalone Content Engine, you must configure the RealMedia players (RealPlayer or RealOne player) on client desktops to point directly to this Content Engine (a nontransparent RealProxy server for these web clients).

To explicitly configure the RealMedia player (RealPlayer Version 8.02 or 9.0) on the client desktops to point directly to a specific standalone Content Engine as their RTSP proxy server, follow these steps:

Step 1  Open RealPlayer on the client desktop.

Step 2  From the RealPlayer menu, choose **View > Preferences**.

Step 3  Click the **Proxy** option under Category settings.

Step 4  Click **Change Settings** under Streaming Settings.

Step 5  Click the **Use proxies** radio button.
Step 6  In the RTSP Proxy address field, enter the IP address of the standalone Content Engine that you have configured for RealMedia proxy caching.

Step 7  In the Port field, enter the port number on which the Content Engine will accept RTSP incoming requests.

By default, the Content Engine listens on port 554 for incoming RTSP requests. If you have changed this default port setting on the Content Engine (that is, you have used the `rtsp port incoming port-number` global configuration command to configure the Content Engine to listen for incoming RTSP requests on a port other than port 554), make sure that you enter that port number into the Port field of the Configure Protocol window.

Step 8  Click OK.

---

### Configuring WCCP Routers for Transparent Redirection

Remember that after you configure a standalone Content Engine for transparent proxy caching, you must configure the WCCP Version 2-enabled routers to intercept and redirect content requests transparently to this Content Engine. For information about how to configure the necessary WCCP service on a WCCP Version 2-enabled router, see Table 4-6.

#### Table 4-6  Configuring WCCP Routers to Support Transparent Proxy Routing of Content Requests

<table>
<thead>
<tr>
<th>Transparent Caching Using WCCP</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP transparent caching</td>
<td>Configuring the Standard Web-Cache Service (Service 0) on a Router</td>
</tr>
<tr>
<td></td>
<td>Configuring the Custom-Web-Cache Service (Service 98) on a Router</td>
</tr>
<tr>
<td>HTTP reverse proxy caching</td>
<td>Configuring the Reverse-Proxy Service (Service 99) on a Router</td>
</tr>
<tr>
<td>WMT transparent caching with</td>
<td>Configuring the RTSP Service (Service 80) on a Router</td>
</tr>
<tr>
<td>WMT RTSP transparent redirection</td>
<td>Configuring the WMT-RTSPU Service (Service 83) on a Router</td>
</tr>
<tr>
<td>RealMedia transparent caching</td>
<td>Configuring the RTSP Service (Service 80) on a Router</td>
</tr>
</tbody>
</table>

### Verifying the Basic Configuration

This section provides an example of how to verify the basic configuration on a standalone Content Engine. This is an example of how to verify the configuration of the web-cache service (HTTP transparent caching through WCCP) on a standalone Content Engine. In this example, the following assumption applies:

- There is a single standalone Content Engine (Content Engine A) that has WCCP Version 2 enabled on it.
- There is a single WCCP Version 2-enabled router (Router A) that has been configured to redirect HTTP requests to Content Engine A (transparent proxy server).
- The HTTP transparent caching service has been enabled on Content Engine A, and Content Engine A is configured to accept redirected HTTP requests from Router A.
- The Client A and Client B browsers are not configured to point directly to Content Engine A.
- Client A and Client B are on the same subnet.
To verify that the web-cache service (HTTP transparent caching through WCCP) is working properly, follow these steps:

**Step 1**  
From Client A, use the client browser to open various web pages on the Internet or your intranet. Request the pages more than once. The web servers you connect to must be on a different subnet than Client A, so that the HTTP requests that the Client A browser issues are routed to Router A.

**Step 2**  
Use a login account that has administrator privileges (privilege level of 15) to log in to the Content Engine CLI on Content Engine A.

For more information about how to log in to a standalone Content Engine through the Content Engine CLI, see the “Using Telnet or a Console Session to Log in to a Standalone Content Engine” section on page 4-50.

**Step 3**  
From Content Engine A, display the HTTP caching saving statistics for this Content Engine.

```
ContentEngineA# show statistics http savings
```

```
Statistics - Savings
Requests                       Bytes
---------------------------------------------
Total:                        525980242   79047534484
Hits:                         1966223    19865155481
Miss:                        524014019   59182379003
Savings:                     0.4 %        25.1 %
```

**Tip**  
You can also display these statistics by choosing **Reporting > Savings** from the Content Engine GUI. For information about how to log in to the Content Engine GUI, see the “Logging in to the Content Engine GUI” section on page 4-55.

**Step 4**  
From Content Engine A, display the number of HTTP requests that this Content Engine has received.

```
ContentEngineA# show statistics http requests
```

**Step 5**  
From Client B, use the browser to request the same web pages that you just requested from Client A.  
This step allows you to check whether Content Engine A is storing a copy of the requested web pages in its local cache instead of retrieving the web pages again from the origin web servers.

- The number of cache hits displayed in the output of the **show statistics http savings** command should increase as you use the Client B browser to request the same web pages that you just requested from Client A.
- The number of HTTP requests displayed in the output of the **show statistics http requests** command should increase as you use the Client B browser to request the same web pages that you just requested from Client A.

**Step 6**  
On Router A, open a console or Telnet session.

**Step 7**  
On Router A, display statistics and status information for Router A.

```
RouterA# show ip wccp
```

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

**Step 8**  
Check to see if Router A shows that packets are being redirected to Content Engine A.
• If Router A shows that there are packets being redirected to Content Engine A, the service (transparent redirection of HTTP requests) is operating properly on Content Engine A and Router A.

• If Router A shows that no packets are being redirected to Content Engine A, the web cache service is not operating properly. In this case, you should troubleshoot the problems with your configuration of the web cache service. The following are some examples of how to do this.

  – From Content Engine A, display the list of WCCP services that are currently configured on Content Engine A. See if the standard web-cache service (Web Cache) is listed. Partial sample output is shown here in this display example:

    ContentEngineA# show wccp services
    Services configured on this Content Engine
    Web Cache
    RTSP
    FTP
    ContentEngineA#

  – From Content Engine A, display a list of WCCP-enabled routers that recognize Content Engine A. Partial sample output is shown here in this display example:

    ContentEngineA# show wccp routers
    Routers Seeing this Content Engine
    Router Id    Sent To
     10.0.0.0     10.1.1.1
    Routers not Seeing this Cache Engine
     10.1.1.1
    Routers Notified of but not Configured
     -NONE-

    Check the command output to determine if Router A is on the list of WCCP-enabled routers that recognize Content Engine A.

  – From Content Engine A, display WCCP generic routing encapsulation (GRE) packet-related information for Content Engine A.

    ContentEngineA# show wccp gre

    Check the command output to view the number of redirected packets that Content Engine A has rejected and accepted. See if the number of accepted packets is increasing as you continue to request web pages that are on web servers located on different subnets than the requesting client (Client A and Client B).

  For information about how to use the ACNS software logs, see Chapter 21, “Monitoring Standalone Content Engines and Transactions.”
Modifying the Basic Configuration Through the Setup Utility

If you have previously run the Setup utility on a Content Engine, then when you subsequently launch the Setup utility manually (using `setup` privileged EXEC command), the current basic configuration for the standalone Content Engine appears. You can quickly change the current basic configuration through the Setup utility by entering `y` when prompted, as shown here in this display example:

```
Here is the current profile of this device
CDM device
No
HTTP Proxy Caching        : Yes
HTTP Transparent Caching  : Yes
HTTP Reverse Proxy Caching: Yes
WMT Proxy Caching         : Yes
WMT Transparent Caching   : Yes
Real Media Proxy Caching  : Yes
Real Media Transparent Caching: Yes
Do you want to change this (y/n) [n]: y
```

Logging in to Standalone Content Engines

This section provides an overview of how to use any of the following methods to log in to a standalone Content Engine:

- Using Telnet or a Console Session to Log in to a Standalone Content Engine, page 4-50
- Using Secure Shell Version 1 or Version 2 to Log in to a Standalone Content Engine, page 4-52
- Using the Content Engine GUI to Log in to a Standalone Content Engine, page 4-53

The ACNS software comes with a single predefined login account (root administrator) that can be used to access the Content Engine initially, and then to add other users to the system. This login account has superuser privilege rights (privilege level of 15). The username for this predefined login account is admin and the default password is default. If these defaults have been changed by another ACNS system administrator, you must obtain the new username and password.

Using Telnet or a Console Session to Log in to a Standalone Content Engine

To log in to a standalone Content Engine using Telnet or a console session, follow these steps:

**Step 1** Log in to the Content Engine using Telnet or a console connected to the Content Engine serial port. For example, after starting a Telnet session, use the `open` command to specify the Content Engine that you want to log in to:

```
Microsoft Telnet> open IP_address_of_Content_Engine
```

**Step 2** When prompted for a login, enter a username and password. (See Figure 4-15.)
Step 3  After you have successfully logged in, the Content Engine CLI displays one of the following prompts depending on the privilege level of the login account:

- Privileged EXEC mode (privilege level of 15):
  
  ContentEngine#

- User EXEC mode (privilege level of 0):
  
  ContentEngine>

where ContentEngine is the host name of the Content Engine.

Figure 4-16 shows an example of the system prompt for privileged EXEC mode.

Tip  A Telnet session with the Content Engine can remain open and inactive for the interval of time specified by the exec-timeout global configuration command. The default timeout is 15 minutes; valid values are 0–44640 minutes. When the exec-timeout interval elapses, the Content Engine automatically closes the Telnet session.

Step 4  Use the Content Engine CLI commands to launch the Setup utility (enter the setup privileged EXEC command) or to use CLI commands to configure or monitor this standalone Content Engine.

See the “ACNS Software CLI Command Modes for Standalone Content Engines” section on page B-8 for a description of the different command modes you can work in when using the ACNS software CLI to configure or monitor a standalone Content Engine. See the “ACNS Software CLI Online Help and Keyboard Shortcuts” section on page B-10 for information about using online help and keyboard shortcuts.
Note The ACNS software device mode determines whether the device is functioning as a Content Engine, Content Distribution Manager, Content Router, or IP/TV Program Manager. The commands available from a specific CLI mode are determined by the ACNS software device mode in effect. The default device operation mode is Content Engine.

Step 5 Enter the logout or exit EXEC commands to end the CLI session at any time.

A Telnet session with the Content Engine can remain open and inactive for the interval of time specified by the exec-timeout global configuration command. The default timeout is 15 minutes; valid values are 0–44, or 640 minutes. When the exec-timeout interval elapses, the Content Engine automatically closes the Telnet session.

Using Secure Shell Version 1 or Version 2 to Log in to a Standalone Content Engine

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 global configuration command to generate a private and a public host key, which the client programs use to verify the 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 shows how to generate 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)# ssd enable
Starting ssh daemon..
Ssh daemon started successfully
```
Secure File Transfer Protocol Access for Nonadministrative Users

In the ACNS 5.3.5 software release, the Secure File Transfer Protocol (SFTP) server on the Content Engine was enhanced to allow nonadministrative users (that is, a user with a nonzero UID) to use SFTP to access the Content Engine. In the ACNS 5.3.5 software release, the `sshd allow-non-admin-users` and `no sshd allow-non-admin-users` CLI global configuration commands were added to enable and disable this new feature. By default, this feature is disabled on the Content Engine, and nonadministrative users cannot use SFTP to access the Content Engine. To enable this feature, enter the `sshd allow-non-admin-users` command on the Content Engine. After enabling this feature, you can disable it again by entering the `no sshd allow-non-admin-users` command on the Content Engine.

If this feature is enabled, the output of the `show running-config` EXEC command shows that this feature is enabled on the Content Engine.

Using the Content Engine GUI to Log in to a Standalone Content Engine

The Content Engine GUI (Figure 4-17) is the web portal for configuring a standalone Content Engine as a caching and streaming engine.

```
Figure 4-17  Content Engine GUI
```

The Content Engine GUI provides access to all of the administrative and operator functions that are accessible to the specific user logged in to the GUI.

After the ACNS software is installed on the Content Engine, use a standard web browser to log in and access the Content Engine GUI.
Before logging in to the Content Engine GUI, check that you have the following information:

- Name or IP address of the Content Engine that you want to log in to.
- Login account (username and password) that you want to log in to the Content Engine for configuration, monitoring, or troubleshooting purposes. If you do not have a login account, your ACNS system administrator must create one for you.
- Type of access enabled on the Content Engine GUI (secure or nonsecure).

**Note** For more information about enabling secure or nonsecure access to the Content Engine GUI, see the “Enabling or Disabling Access to the Content Engine GUI” section on page 4-54.

### Enabling or Disabling Access to the Content Engine GUI

You can configure secure or nonsecure access to the Content Engine GUI. Secure access is the default. Either secure or nonsecure access to the Content Engine GUI is possible but not both. For example, if the secured Content Engine GUI is enabled (for example, https:// access on port 8003), then nonsecure access to the Content Engine GUI (for example, http:// access on port 8001) is not allowed.

- To enable or specify the port number of the Content Engine GUI server, use the `gui-server` global configuration command.

  ```
  ContentEngine(config)# gui-server {enable | port port | secure {enable | port port}}
  ```

- To enable secure access to the Content Engine GUI, use the `gui-server secure enable port` global configuration command.

  ```
  ContentEngine(config)# gui-server secure enable port 8003
  ```

  The port number can be between 1 and 65535. The default port for secure access to the GUI is 8003. In this example, secure access to the Content Engine GUI is enabled on the default port number 8003.

- To enable nonsecure access to the Content Engine GUI, use the `gui-server enable port` global configuration command.

  ```
  ContentEngine(config)# gui-server enable port 8001
  ```

  The port number can be between 1 and 65535. The default port for nonsecure access to the GUI is 8001. In this example, nonsecure access to the Content Engine GUI is enabled on the default port number 8001.

**Note** When secure or nonsecure access to the Content Engine GUI is enabled, you can access the GUI as described in the “Logging in to the Content Engine GUI” section on page 4-55.

To disable the Content Engine GUI, use the `no gui-server` global configuration command:

```
no gui-server {enable | port | secure {enable | port port}}
```

For example, if secure access to the Content Engine GUI is enabled on port 8003, enter the following command to disable it:

```
ContentEngine(config)# no gui-server secure enable port 8003
```
In the following example, nonsecure access to the Content Engine on port 8001 is being disabled:

```
ContentEngine(config)# no gui-server enable port 8001
```

## Logging in to the Content Engine GUI

To log in to the Content Engine GUI, follow these steps:

---

**Step 1**

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

**Tip**

If you are using Microsoft Internet Explorer (IE), verify that Java, JavaScript, and Cascading Style Sheets are enabled on IE. If you are using Netscape, use Version 4.0 or later.

**Step 2**

In the web browser, enter the URL or IP address of the Content Engine. Append the port number.

The URL (location) of the Content Engine is determined during the installation of the ACNS software. If your network supports DNS and the IP address of the Content Engine has been added to your DNS table, you can access the Content Engine GUI by using the DNS name of the Content Engine.

The port number of the Content Engine GUI is determined when the ACNS software is installed on the Content Engine. The default port number for nonsecure access is 8001. The default port number for secure access is 8003. Secure access to the Content Engine GUI is enabled on the default port number 8003. HTTP is used for nonsecure access and HTTPS is used for secure access.

The following example shows the URL for accessing the Content Engine GUI in nonsecure mode if nonsecure mode is enabled on the default port (port 8001):

```
http://ContentEngine-name:8001
```

Alternatively, enter the IP address:

```
http://ContentEngine-IP-address:8001
```

The following example shows the URL for accessing the Content Engine GUI in secure mode if secure mode is enabled on the default port (port 8003):

```
https://ContentEngine-name:8003
```

Alternatively, enter the IP address:

```
https://ContentEngine-IP-address:8003
```

**Step 3**

If you specified secure access, then the Security Alert window appears. Click Yes to accept the security certificate. The Enter Network Password window appears.

**Step 4**

Enter your username in the Username field. Enter your password in the Password field and click OK.

Username: admin
Password: password

**Step 5**

After the system verifies the specified login information, the main window for the Content Engine GUI appears in your browser. If you are the default administrator, you should create login accounts for your ACNS system administrators or other ACNS administrative who need to access the Content Engine for configuration, monitoring, or troubleshooting purposes.

When you access the Content Engine GUI, it appears with a window (page) that is referred to as the Content Engine main window. (See Figure 4-17.) The Content Engine GUI has a set of tabs and buttons. For a descriptive list of menu options, see Appendix A, “Content Engine GUI Menu Options.”
Tip

The lock icon in the lower-right corner of the browser window indicates that the Content Engine GUI has been accessed in secure mode instead of nonsecure mode. For information about enabling secure or nonsecure access to the Content Engine GUI, see the “Enabling or Disabling Access to the Content Engine GUI” section on page 4-54.

Table 4-7 describes the main buttons of the Content Engine GUI and their associated function.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>Removes all cacheable objects from the Content Engine memory and hard disks.</td>
</tr>
<tr>
<td>Reboot</td>
<td>Reboots the Content Engine.</td>
</tr>
<tr>
<td>Save Config</td>
<td>Saves the running system configuration to the startup system configuration.</td>
</tr>
<tr>
<td>Update</td>
<td>Applies the changes specified in the current Content Engine GUI window to</td>
</tr>
<tr>
<td></td>
<td>the running system configuration.</td>
</tr>
<tr>
<td>Help</td>
<td>Displays context-sensitive help for the particular Content Engine GUI window.</td>
</tr>
<tr>
<td></td>
<td>Click the Back button in the Help window to return to the Content Engine</td>
</tr>
<tr>
<td></td>
<td>GUI window from which you launched the context-sensitive help.</td>
</tr>
</tbody>
</table>

Step 6 To return to the main window, click the words Content Engine in the upper-left corner of any Content Engine window (see Figure 4-18).

Figure 4-18 Content Engine GUI Navigation Bar

Step 7 To navigate to another window in the Content Engine GUI, click one of the tabs to display its subtabs. Click a subtab to choose it (see Figure 4-19).
Figure 4-19  Content Engine GUI Tabs and Subtabs

Logging out of the Content Engine GUI

To log out of the Content Engine GUI, follow these steps:

**Step 1**  Click the **Update** button to save any changes that you made in the current Content Engine window, or click **Cancel** to cancel these changes.

**Step 2**  Return to the Content Engine main window (Figure 4-17) by clicking the words Content Engine in the upper-left corner of the current Content Engine window.

**Step 3**  To save any changes made during this current session before logging out, click the **Save Config** button in the Content Engine main window.

**Step 4**  Choose **File > Close**, or click the browser **Close** button.
Performing Other Basic Tasks for Standalone Content Engines

After you have done a basic configuration on a standalone Content Engine, you can perform other basic tasks such as setting the system clock, managing login accounts, managing and monitoring disks. This chapter describes how to use the Content Engine CLI to perform the following basic tasks for standalone Content Engines:

- Showing Inventory, page 5-2
- Managing Administrative Login Accounts, page 5-3
- Setting the System Clock, page 5-4
- Configuring Banners for Standalone Content Engines, page 5-6
- Adding or Modifying Administrative Login Accounts, page 5-8
- Configuring Disk Space, page 5-9
- Removing All Disk Partitions on a Single Disk Drive, page 5-11
- Stopping Applications from Using a Disk Drive, page 5-12
- Displaying the Current Disk Configuration, page 5-12
- Mounting to a Network Attached Storage Device, page 5-13
- Saving the Current Configuration on Standalone Content Engines, page 5-13
- Disabling Transparent Caching Services on Standalone Content Engines, page 5-14
- Creating Custom Message Pages for Standalone Content Engines, page 5-14
- Removing or Replacing a Content Engine, page 5-24
- Remotely Upgrading the BIOS, page 5-24

**Note**

For information about how to log in to the Content Engine, see the “Logging in to Standalone Content Engines” section on page 4-50.
Cisco Content Engines are embedded with following three identification items:

- Product ID (PID)
- Version ID (VID)
- Serial number (SN)

This identity information is stored in non-volatile memory. Each Content Engine has a unique device identifier (UDI). The UDI = PID + VID + SN.

The UDI is electronically accessed by the product operating system or network management application to enable identification of unique hardware devices. Consequently, the data integrity of the UDI is vital to customers. This means that the UDI that is programmed into the Content Engine’s non-volatile memory is equivalent to the UDI that is printed on the product label and on the carton label. This UDI is also equivalent to the UDI that can be viewed through any electronic means, and in all customer-facing systems and tools.

In the ACNS 5.2.1 software and later releases, enter the `show inventory` EXEC command to view the Content Engine’s UDI. Currently, there is only CLI access to the UDI; there is no SNMP access to the UDI information.

All of the Content Engine models that are supported in the ACNS 5.2.1 software and later releases support this show inventory feature.

On newer Content Engine models, you can display the Content Engine’s UDI by using a single command, the `show inventory` EXEC command.

```plaintext
CE-565# show inventory
PID: CE-565-K9 VID: 0 SN: serial number
serial number is the serial number of the Content Engine. If the version number is available, then it is displayed; otherwise, a zero (0) is displayed (as shown in the example).
```

On older Content Engine models (for example, the CE-507 or CE-2636), you must use the `show tech-support` and `show inventory` EXEC commands to display the Content Engine’s UDI.

```plaintext
CE-507# show inventory
Please look at 'sh tech-support' for information!
CE-507# show tech-support
```

**Note**

See the *Release Notes for Cisco ACNS Software, Release 5.5* for a list of hardware platforms that are supported in the ACNS 5.5 software release.
Managing Administrative Login Accounts

A Content Engine that is running the ACNS software comes with a single predefined superuser login account (root administrator). This predefined administrative login account can be used to access the Content Engine GUI initially in order to perform a basic configuration on a standalone Content Engine and then add other login accounts.

The username for this predefined superuser account is admin and the default password is default. If these defaults have been changed by another ACNS system administrator, you need to obtain the new username and password.

You must assign a privilege profile to each new administrative login account that you create on your standalone Content Engine. Privilege profiles determine which tasks ACNS software administrators can perform, and the level of access granted to them based on the administrative login account that they used to log in to the Content Engine.

The following are the two types of predefined privilege profiles:

- Normal-level administrator—Privilege level of zero (0). Has read access, and can see some of the Content Engine configuration settings.
- Superuser administrator—Privilege level of 15. Has administrative privileges such as running the Setup utility, creating new administrative login accounts, and modifying any of the Content Engine configuration settings.

You can use the Content Engine GUI or CLI to change the password for this predefined superuser account or to create additional login accounts for other ACNS system administrators. After you have used the predefined superuser login account to perform a basic configuration on a standalone Content Engine, it is recommended that you change the password for this superuser login account. For information about adding or modifying an administrative login account, see the next section, “Setting the System Clock.”

The ACNS 5.x software also supports various login authentication methods (local, RADIUS, or TACACS+ authentication). This enables you to configure a standalone Content Engine to use one or more of these authentication methods when it processes an administrative login request. For more information, see Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines.”

Content authentication and authorization, which controls end users’ access to the requested content that is served through a standalone Content Engine, is independent of the administrative login authentication and authorization for the Content Engine that controls the level of access for users who log in to the Content Engine for configuration, monitoring, or troubleshooting purposes. For information about content authentication and authorization, see Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.”
Setting the System Clock

If you have an outside source on your network that provides time services (such as a Network Time Protocol [NTP] server), you do not need to set the system clock manually. When setting the clock, enter the local time. The Content Engine calculates Coordinated Universal Time (UTC) based on the time zone specified in the `clock timezone` global configuration command.

Two clocks exist in the system: the software clock and the hardware clock. The software uses the software clock. The hardware clock is used only at bootup to initialize the software clock.

To set or clear clock functions or update the calendar, use the `clock` EXEC command.

```
clock {read-calendar | set time day month year | update-calendar}
```

where:

- **read-calendar**: Reads the calendar and updates the system clock.
- **set**: Sets the time and date of the software clock.
- **time**: Current time in hh:mm:ss format (hh: 00–23; mm: 00–59; ss: 00–59).
- **day**: Day of the month (1–31).
- **month**: Month of the year (January, February, March, April, May, June, July, August, September, October, November, December).
- **year**: Year (1993–2035).
- **update-calendar**: Updates the calendar with the system clock.

For example:

```
ContentEngine# clock set 13:32:00 01 February 2000
```

To set the daylight saving time and time zone for display purposes, use the `clock` global configuration command. To set and display the local and UTC current time of day without an NTP server, use the `clock timezone` global configuration command with the `clock set` EXEC command. The `clock timezone` parameter specifies the difference between UTC and local time, which is set with the `clock set` EXEC command. The UTC and local time are displayed with the `show clock detail` EXEC command.

An accurate clock and timezone setting is required for the correct operation of HTTP proxy caches.

Displaying the Standard Time Zones

The ACNS system has several predefined standard time zones. Some of these time zones have built-in daylight saving time information while others do not. For example, if you are in an eastern region of the United States (US), you must use US/Eastern time zone, which includes daylight saving time information and will adjust the clock automatically every April and October. The system includes about 1500 standard time zone names.

In the ACNS 5.2.x software and earlier releases, there was no restriction on these reserved standard time zone names. You could overload these standard names in various ways. For example, you could use the US/Pacific time zone but entering the `clock summertime` EXEC command to define a different daylight saving time schedule. In the ACNS 5.3.1 software and later releases, strict checking is supported. The `clock summertime` command is now disabled when a standard time zone is configured. You can only configure daylight saving time if the time zone is a customized zone (that is, not a standard time zone).

The `show clock standard-timezones all` EXEC command allows you to browse through all standard timezones and choose from these predefined time zones.
You can choose a customized name that does not conflict with the predefined names of the standard time zones. Most predefined names of the standard time zones have two components, a region name and a zone name. You can list time zones by several criteria, such as regions and zones.

\textbf{show clock standard-timezones \{all \ | \ regions \ | \ zones region name \ | \ details complete name \}}

The following example shows a portion of the output from the \textbf{show clock standard-timezones all} EXEC command. As the following example shows, all of the standard time zones (approximately 1500 time zones) are listed. Each time zone is listed on a separate line.

\begin{verbatim}
ContentEngine # show clock standard-timezones all
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera

The following example shows a portion of the output from the \textbf{show clock standard-timezones region} EXEC command. As the example shows, all first level time zone names or directories are listed. All 1500 time zones are organized into directories by region.

\begin{verbatim}
ContentEngine # show clock standard-timezones regions
Africa/
America/
Antarctica/
Arctic/
Asia/
Atlantic/
Australia/
Brazil/
CET
- 
- 
US/
UTC
Universal
W-SU
WET
Zulu

The following example shows a portion of the output from the \textbf{show clock standard-timezones zones} EXEC command. As the following example shows, this command lists the name of every time zone that is within the specified region (for example, the US region).

\begin{verbatim}
ContentEngine # show clock standard-timezones zones US
Alaska
Aleutian
Arizona
Central
East-Indiana
Eastern
Hawaii
Indiana-Starke
Michigan
Mountain
Pacific
Samoa

The following sample shows a portion of the output from the \textbf{show clock standard-timezones details} EXEC command. As the following example shows, this command shows details about the specified time zone (for example, the US/Eastern time zone). The command output also includes the standard offset from the Greenwich mean time (GMT).
Configuring Banners for Standalone Content Engines

In the ACNS 5.3.1 software and later releases, you can configure the following three types of banners in any ACNS software device mode:

- motd banner—Sets the message of the day. This message is the first message that is displayed when a login is attempted.
- login banner—Displayed after the motd banner but before the actual login prompt appears.
- exec banner—Displayed after the EXEC CLI shell has started.

All three of these banners are effective on a console, Telnet, or a Secure Shell (SSH) Version 2 session. When you run an SSH Version 1 client and log in to the Content Engine, the motd and login banners are not displayed.

The `banner` global configuration commands allow you to configure the different types of banners for standalone Content Engines.

```
banner {motd | login | exec} {message text message | <cr>}
```

The following example shows how to use the `banner motd message` global configuration command to configure the motd banner. In this example, the motd message consists of a single line of text.

```
ContentEngine(config)# banner motd message This is an ACNS 5.4 device
```

The following example shows how to use the `banner motd message` global command to configure a MOTD message that is longer than a single line. In this case, the Content Engine translates the “\n” portion of the message to a new line when the MOTD message is displayed to the user.

```
ContentEngine(config)# banner motd message "This is an ACNS device. \nAccess is restricted. \n"
```

After you configure the banners, enter the `banner enable` global configuration command to enable banner support on the Content Engine. Enter the `show banner` EXEC command to display information about the configured banners.
This example shows how to configure and enable banner support on a standalone Content Engine (Content Engine A). In this example, the user logs in to Content Engine A through an SSH session.

---

### Step 1
Administrator A uses the `banner message` commands to configure the motd, login, and exec banners on Content Engine A, as follows:

**a.** Configure the motd message. In this example, the motd message is longer than a single line. In this case, Content Engine A translates the “\n” portion of the message to a new line in the motd message that is displayed to the user (in this case, Administrator B).

```
ContentEngine (config)# banner motd message "This is the motd message. \nThis is an ACNS 5.4 device\n"
```

**b.** Configure a login message. This example shows how to configure a motd message that is longer than a single line. In this case, Content Engine A translates the “\n” portion of the message to a new line in the login message that is displayed to the user (in this case, Administrator B).

```
ContentEngine(config)# banner login message "This is login banner. \nUse your password to login\n"
```

**c.** Configure an interactive banner. The `banner exec` command is almost identical to the `banner motd` message command except that with the `banner exec` command the banner content is obtained from the command line input that the user enters after being prompted for the input.

```
ContentEngine (config)# banner exec message "This is the EXEC banner. \nUse your ACNS username and password to log in to this Content Engine.\n"
```

### Step 2
Administrator A enables banner support on Content Engine A.

```
ContentEngine (config)# banner enable
```

### Step 3
Administrator A enters the `show banner` EXEC commands to display information about the configured banners. For example, the `show banner motd` EXEC command is used to display information about the configured message of the day (motd) banner.

### Step 4
Another ACNS administrator (Administrator B) uses an SSH session to log in to Content Engine A. For example:

```
% ssh admin@ce
```

After Administrator B logs in to the SSH session, Administrator B will see a login session that includes a motd banner as well as a login banner that asks Administrator B to enter a login password. For example:

```
This is the motd banner.
This is an ACNS 5.4 device
This is login banner.
Use your password to login.
Cisco Content Engine
admin@ce's password:
```

After Administrator B enters a valid login password, the EXEC banner is displayed, and Administrator B is asked to enter the ACNS username and password. For example:

```
Last login: Fri Oct 1 14:54:03 2004 from client
System Initialization Finished.
This is the EXEC banner.
Use your ACNS username and password to log in to this Content Engine.
```
After Administrator B enters a valid ACNS username and password, the Content Engine CLI is displayed. The CLI prompt varies depending on the privilege level of the login account. In this example, because Administrator B entered a username and password that had administrative privileges (privilege level of 15), the EXEC mode CLI prompt is displayed.

ContentEngine# 

### Adding or Modifying Administrative Login Accounts

After you have used the predefined superuser login account to perform a basic configuration on a standalone Content Engine, we recommend that you change the password for this superuser login account.

From the Content Engine GUI, choose System > Users. Use the displayed Users window to modify a password for this predefined superuser login account. You can also use the Users window to create additional administrative login accounts (normal administrative user or superuser accounts). For information about using this window, click the HELP button to access context-sensitive help. For more information about accessing the Content Engine GUI, see the “Logging in to the Content Engine GUI” section on page 4-55.

To use the Content Engine CLI to add or modify an ACNS system administrative login account on a standalone Content Engine, follow these steps:

**Step 1** Access the Content Engine CLI in global configuration mode.

**Step 2** Configure the entries for the group name-based access list.

ContentEngine(config)# username name {cifs-password 0 plainword | 1 lancrypto ntcrypto | password 0 plainword | 1 cryptoword uid uid | privilege {0|15}}

Table 5-1 describes the parameters for the username global configuration command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Sets the username for the administrative login account.</td>
</tr>
<tr>
<td>name</td>
<td>Username for the administrative login account.</td>
</tr>
<tr>
<td>cifs-password</td>
<td>Sets the Windows file-sharing user password.</td>
</tr>
<tr>
<td>0</td>
<td>Specifies that an unencrypted Windows file sharing password will follow.</td>
</tr>
<tr>
<td>plainword</td>
<td>Clear-text user Windows file sharing password.</td>
</tr>
<tr>
<td>1</td>
<td>Specifies that a hidden Windows file sharing password will follow.</td>
</tr>
<tr>
<td>lancrypto</td>
<td>Encrypted password for LAN Manager networks.</td>
</tr>
<tr>
<td>ntcrypto</td>
<td>Encrypted password for Windows NT networks.</td>
</tr>
<tr>
<td>password</td>
<td>Sets the user password for the administrative login account.</td>
</tr>
<tr>
<td>0</td>
<td>Specifies that an unencrypted user password will follow.</td>
</tr>
<tr>
<td>1</td>
<td>Specifies that a hidden user password will follow.</td>
</tr>
<tr>
<td>cryptoword</td>
<td>Encrypted user password.</td>
</tr>
<tr>
<td>uid</td>
<td>Sets the user ID for the password.</td>
</tr>
</tbody>
</table>
This example shows how you can use the `username` EXEC command to modify passwords and privilege levels for administrative login accounts on a standalone Content Engine:

```
ContentEngine# show user username jrdoe
Uid : 2003
Username : jrdoe
Password : ghQ.GyGhP96K6
Privilege : normal user

ContentEngine(config)# username jrdoe privilege 15
User's privilege changed to super user (=15)

ContentEngine# show user username jrdoe
Uid : 2003
Username : jrdoe
Password : ghQ.GyGhP96K6
Privilege : super user
```

### Configuring Disk Space

Disk space in ACNS software is allocated on a per-file system basis, rather than on a per-disk basis. You can configure your overall disk storage allocations according to the kinds of client protocols you expect to use and the amount of storage that you need to provide for each of the functions, as described in Table 5-2. Use the `disk add` EXEC command to add a single disk with the specified partitions.

#### Table 5-2 Cisco ACNS Software Disk Storage for Standalone Content Engines

<table>
<thead>
<tr>
<th>Disk Storage Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysfs (system file system)</td>
<td>Stores log files, including transaction logs, syslogs, and internal</td>
</tr>
<tr>
<td></td>
<td>debugging logs. Also can store image files and configuration files.</td>
</tr>
<tr>
<td></td>
<td>For more information about sysfs, see the next section, “Creating Disk</td>
</tr>
<tr>
<td></td>
<td>Space for the Sysfs.”</td>
</tr>
<tr>
<td>cfs (cache file system)</td>
<td>Caches HTTP and FTP objects.</td>
</tr>
<tr>
<td>mediafs (media file system)</td>
<td>Caches content from streaming proxy servers, such as WMT and RealProxy.</td>
</tr>
</tbody>
</table>
Enter the `show disks` EXEC command to view information about the current disk configuration of a standalone Content Engine.

```
ContentEngine# show disks
Local disks:
  SYSFS                29.9GB      96.8%
  CFS                   0.0GB       0.0%
  MEDIAFS               0.0GB       0.0% (from-unused-cdnfs)
  CDNFS                 1.0GB       3.1%
  FREE                  0.0GB       0.0%
```

Note: CDNFS and MEDIAFS amounts are reported in terms of actual usable amounts of storage for applications. Due to internal filesystem overhead of approximately 3%, the reported amounts may be smaller than what you configured. CDNFS space is allocated with higher priority than MEDIAFS, so if you configured MEDIAFS and CDNFS, then MEDIAFS will be reduced by the amount of the total CDNFS and MEDIAFS overhead. If you have not configured MEDIAFS, then CDNFS will be reduced by the amount of the overhead.

```
Network-attached disks:
  NONE
ContentEngine#
```

**Note**

Standalone Content Engines do not have a CDNFS partition because this partition is used to store pre-positioned content, which a standalone Content Engine does not support.

Use the `disk config sysfs` EXEC command to configure disk resources for standalone Content Engine.

In the ACNS 5.2.x software and earlier releases, the CDNFS, MEDIAFS, and SYSFS partitions use the ext2 file system. With ext2 file systems, if the system crashed or if the system is not shut down properly a file system check of these partitions takes a long time. If there are sector failures on the disk, the time to perform a file system check with an ext2 file system increases even more. In the ACNS 5.3.1 software and later releases, the ext3 file system is used instead of the ext2 file system. By migrating to the ext3 file system, the amount of time required to perform a file system check of the CDNFS, MEDIAFS, and SYSFS partitions is decreased, which increases the availability of the Content Engine. If you are upgrading from an earlier release of the ACNS software, the ext2 file system is automatically converted to the ext3 file system when you upgrade to the ACNS 5.3.1 software and later releases.

In the ACNS 5.2.1 software and later releases, the ability to monitor Content Engine disk drives is supported. Disk status is now recorded in flash (non-volatile storage). When an error on a Content Engine disk device occurs, a message is written to the system log (syslog) if the sysfs partition is still intact, and an SNMP trap is generated if SNMP is configured on the Content Engine.

In addition to tracking the state of critical disk drives, you can define a disk device error-handling threshold on the Content Engine. If the number of disk device errors reaches the specified threshold, the corresponding disk device is automatically marked as “bad.” The ACNS system does not stop using the bad disk device immediately; it stops using the bad disk drive after the next reboot.

If the specified threshold is exceeded, the Content Engine either records this event or reboots. If the automatic reload feature is enabled and this threshold is exceeded, then the ACNS system automatically reboots the Content Engine. For more information about specifying this threshold, see the “Specifying the Disk Error-Handling Threshold” section on page 21-18. For more information about monitoring critical disks, see the “Monitoring Critical Disk Drives on Standalone Content Engines” section on page 21-17.
In the ACNS 5.3.1 software and later releases, the ability to monitor proactively the health of disks with Self Monitoring, Analysis, and Reporting Technology (SMART) is supported. SMART provides you with hard drive diagnostic information and information about impending disk failures. For more information, see the “Proactively Monitoring Disk Health with SMART” section on page 21-20.

 Creating Disk Space for the Sysfs

If you are initially configuring a standalone Content Engine, you must create disk space for the system file system (sysfs) by using the **disk config sysfs** EXEC command.

To configure disk space on a standalone Content Engine, follow these steps:

---

**Step 1** Exit configuration mode, if you have not already done so.

```
ContentEngine(config)# exit
ContentEngine#
```

**Step 2** Configure the disk space for the sysfs. For example, to configure the sysfs for 5 GB, enter this command:

```
ContentEngine# disk config sysfs 5GB
```

**Step 3** Reload the Content Engine for the disk configuration to take effect.

```
ContentEngine# reload
```

Tips For the new disk space configuration to take effect, you must first reboot the software. If you encounter an error message, reenter your disk configuration and use the **reload** EXEC command on the Content Engine for the disk configuration to be applied.

---

 Removing All Disk Partitions on a Single Disk Drive

Use the **disk delete-partitions** EXEC command to remove all disk partitions on a single disk drive.

**Caution** The disk delete-partitions EXEC command will erase everything on the specified disk.

Typically, this command is used when you want to add a new disk drive that was previously used with another operating systems (for example, a Microsoft Windows or Linux operating system). When asked if you want erase everything on the disk, specify **yes** to proceed:

```
ContentEngine# disk delete-partitions disk03
This will erase everything on disk. Are you sure? [no] yes
```
Stopping Applications from Using a Disk Drive

In the ACNS 5.3.1 software and later releases, the **disk unuse** EXEC command allows you to stop applications from using a specific disk drive (for example, disk01) without having to reboot the drive:

```
ContentEngine# disk unuse disk01
```

The disk unuse feature cannot be used with disk00 (the first disk drive) or with the drive that contains the `/local/local1` directory (for example, if disk01 contains the `/local/local1` directory then you cannot use the disk unuse command with disk01). For more information, see the *Cisco ACNS Software Upgrade and Maintenance Guide, Release 5.x*.

Displaying the Current Disk Configuration

Use the **show disks** EXEC commands to display the current disk configuration of a standalone Content Engine.

```
ContentEngine# show disks ?
canfigured        Show configured configurations (effective after reboot)
current           Show current configurations
details           Show current configurations with more details
failed-sectors     Show the list of failed sectors on the disks
network-attached   Show Network Attached Storage on this device
SMART-info        Show hard drive diagnostic and failure prediction info
                    (provided by SMART disk monitor)
storage-array      Show Disk Information on storage array (if any)
```

For example, enter the **show disks details** EXEC command to display detailed information about a Content Engine’s current disk configuration.

```
ContentEngine# show disks details
disk00: Normal          (IDE disk)                 38160MB( 37.3GB)
disk00/04: PHYS-FS       15137MB( 14.8GB) mounted internally
disk00/04: MEDIAFS       532MB(  0.5GB) mounted internally
disk00/05: SYSFS         1023MB(  1.0GB) mounted at /local1
disk00/06: CFS           15359MB( 15.0GB)
System use:               6130MB(  6.0GB)
FREE:                      16MB(  0.0GB)
disk01: Not present
No NAS share is attached to this device.
```

Disk drives that are currently marked as “bad” are shown as “Not used” in the output of the **show disks details** EXEC command. Future “bad” disk drives (drives that will not be used after the next time the Content Engine is reloaded) are shown with an asterisk (*). In the following case, disk03 is a future bad disk drive that will not be used after the Content Engine is reloaded.

```
ContentEngine# show disks details
(*) Disk drive won't be used after reload.
```

```
disk03: Normal        (h00 c00 i03 l00 - Int DAS)        70001MB( 68.4GB) (*)
FREE:                  70001MB( 68.4GB)
```

For information about disk monitoring, see the “Monitoring Critical Disk Drives on Standalone Content Engines” section on page 21-17.
Mounting to a Network Attached Storage Device

The ACNS 5.x software provides a Common Internet File System (CIFS) client and a Network File System (NFS) client for Content Engines to communicate with network-attached storage (NAS) devices. Content Engines can be attached to NAS devices to increase their storage space. These Content Engines function as NFS or CIFS clients while accessing the NAS servers. NAS servers include UNIX-mode NFS servers or Microsoft Windows systems for CIFS sharing.

NAS servers support the cdnfs and mediafs for Content Engines. You can choose the type of file system to be attached to the NAS depending on whether you need to store cached WMT, RealMedia, or other streaming content.

NFS and CIFS servers export either an entire file system to a Content Engine or a specified directory on a file system. In both cases, you need to specify the amount of disk space to be assigned to the Content Engine. Different Content Engines attach different directories on an NFS or CIFS server, and it is not possible to share the same directory among multiple Content Engines. NFS servers support host-based authentication and UNIX file system access control. You need to specify the client IP address that matches the list of hosts that an NFS server trusts. The client is then allowed to mount and access files based on the permissions assigned to it. On the other hand, CIFS servers share files and authenticate users on the server itself, instead of exporting data to clients for authentication. CIFS servers support NTLM, plain text password, and LDAP authentication.

Mounting NAS shares to a Content Engine can be performed in these ways:

- Through the CLI for standalone Content Engines
- Through the Content Distribution Manager GUI or the CLI for centrally managed Content Engines

In the ACNS 5.3.1 software and later releases, you can configure a CIFS share name by using the share-name option of the network-filesystem server cifs share-web-site global configuration command. For more information about how to mount a Content Engine to a NAS device and configure a CIFS share name through the CLI, see the Cisco ACNS Software Upgrade and Maintenance Guide, Release 5.x.

Saving the Current Configuration on Standalone Content Engines

A standalone Content Engine has two types of system configuration:

- Startup system configuration that is stored in nonvolatile memory
- Running system configuration

To use the Content Engine CLI to save the current running configuration as the startup configuration, use the copy running-config global configuration command. The running system configuration can be saved to the sysfs partition, flash memory, or TFTP server. For example, enter the following to save the running configuration to flash memory:

```
ContentEngine (config)# copy running-config startup-config
```

Note

The copy running-config startup-config command is equivalent to the write memory command.

To save the current configuration during a Setup utility session, enter y when asked if you want to save the current configuration.
Disabling Transparent Caching Services on Standalone Content Engines

To disable transparent caching on a standalone Content Engine in a WCCP environment without powering down the Content Engine, disable the running version of WCCP on the Content Engine by entering the `no wccp version` global configuration command (for example, use the `no wccp version 2` command to disable WCCP Version 2). The Content Engine will still service proxy-style requests, if so configured, and preserve its configuration settings.

Creating Custom Message Pages for Standalone Content Engines

In the ACNS 5.4.1 software and later releases, you can create the following types of customized error pages on a standalone Content Engine:

- HTTP custom error message pages, as described in the next section, “Creating HTTP Custom Error Pages for Standalone Content Engines”
- FTP native custom message pages, as described in the “Creating Custom Messages for FTP Proxy Responses for FTP Native Requests” section on page 5-19.

Creating HTTP Custom Error Pages for Standalone Content Engines

In the ACNS 5.1 software and later releases, you can create HTTP customized error pages. If you create these customized pages, then the standalone Content Engine displays the appropriate customized error page instead of the default error message when proxy errors occur for HTTP (HTTP, HTTPS, and FTP-over-HTTP) requests.

Table 5-3 describes the custom HTTP error messages and their usage for standalone Content Engines running the ACNS 5.3.1 software and later releases.

<table>
<thead>
<tr>
<th>Message Identifier</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocked-dueto-filter-error</td>
<td>Error response when a request is blocked because of a filter.</td>
</tr>
<tr>
<td>cache-read-error</td>
<td>Error response when a cache files system (cfs) read fails.</td>
</tr>
<tr>
<td>cache-write-error</td>
<td>Error response when a cfs write fails.</td>
</tr>
<tr>
<td>client-access-denied-msg</td>
<td>Error response when a client access is denied.</td>
</tr>
<tr>
<td>client-connection-broken-error</td>
<td>Error response when a client connection is lost.</td>
</tr>
<tr>
<td>dns-not-available-error</td>
<td>Error response when DNS is unavailable for resolution.</td>
</tr>
</tbody>
</table>
| error-signature          | Signature message that is appended to the final error page that is displayed to the end user (added in the ACNS 5.3.1 software release). If you do not create a customized error-signature message, then the default error-signature message is used. For more information about creating a customized error-signature message, see the “Creating a Custom Error-Signature Message for HTTP Custom Error Pages” section on page 5-17.
### Table 5-3 Custom Error Page Messages for Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Message Identifier</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>expect-failed-error</td>
<td>Error response when Expect specifier in the HTTP request header cannot be met.</td>
</tr>
<tr>
<td>ftp-bad-login-error</td>
<td>Error response when the FTP login fails.</td>
</tr>
<tr>
<td>ftp-bad-url-error</td>
<td>Error response when the FTP request receives a bad URL.</td>
</tr>
<tr>
<td>ftp-disabled-error</td>
<td>Error response when the FTP is disabled.</td>
</tr>
<tr>
<td>ftp-failure-error</td>
<td>Error response when FTP fails.</td>
</tr>
<tr>
<td>ftp-internal-error</td>
<td>Error response when the FTP interval is exceeded.</td>
</tr>
<tr>
<td>ftp-not-found-error</td>
<td>Error response when the FTP file not found.</td>
</tr>
<tr>
<td>ftp-put-created-msg</td>
<td>Error response when the FTP PUT is successful.</td>
</tr>
<tr>
<td>ftp-put-error</td>
<td>Error response when the FTP PUT fails.</td>
</tr>
<tr>
<td>ftp-put-modified-msg</td>
<td>Response when the FTP update is successful.</td>
</tr>
<tr>
<td>ftp-unavailable-msg</td>
<td>Error response when the FTP file is unavailable.</td>
</tr>
<tr>
<td>http-blocked-port-msg</td>
<td>Error response when an HTTP request comes through a blocked port.</td>
</tr>
<tr>
<td>https-blocked-port-msg</td>
<td>Error response when an HTTPS request comes through a blocked port.</td>
</tr>
<tr>
<td>icap-processing-error</td>
<td>Error response when an error occurred in ICAP processing.</td>
</tr>
<tr>
<td>invalid-port-error</td>
<td>Error response when an invalid port is accessed.</td>
</tr>
<tr>
<td>looped-req-error</td>
<td>Error response when a looped request is unsuccessful.</td>
</tr>
<tr>
<td>not-enough-resources-error</td>
<td>Error response when enough resources are not available for the request process.</td>
</tr>
<tr>
<td>not-in-cache</td>
<td>Error response when the object is not found in the cache.</td>
</tr>
<tr>
<td>offline-miss-error</td>
<td>Error response when an off-line Content Engine finds a cache miss.</td>
</tr>
<tr>
<td>outgoing-proxy-fail-error</td>
<td>Error response when all outgoing proxy fails.</td>
</tr>
<tr>
<td>proxy-allow-domain-error</td>
<td>Error response when the domain is not allowed to authenticate in proxy mode (added in the ACNS 5.3.1 software release).</td>
</tr>
<tr>
<td>proxy-no-default-domain-error</td>
<td>Error response when there is no default domain available to authenticate in proxy mode (added in the ACNS 5.3.1 software release).</td>
</tr>
<tr>
<td>proxy-unauthenticated-error</td>
<td>Error response when the proxy authentication fails.</td>
</tr>
<tr>
<td>radius-redirect-error</td>
<td>Error response for a RADIUS redirect message.</td>
</tr>
<tr>
<td>request-blocked-msg</td>
<td>Error response when the request is blocked.</td>
</tr>
<tr>
<td>request-malformed-error</td>
<td>Error response when the request headers are malformed.</td>
</tr>
<tr>
<td>rev-dns-not-available-msg</td>
<td>Error response when DNS is not available.</td>
</tr>
<tr>
<td>server-connection-broken-error</td>
<td>Error response when the server connection is lost.</td>
</tr>
<tr>
<td>www-allow-domain-error</td>
<td>Error response when the domain is not allowed to authenticate (added in the ACNS 5.3.1 software release).</td>
</tr>
</tbody>
</table>
You can use the Content Engine GUI or the CLI to create HTTP customized error pages for any of the error messages listed in Table 5-3.

- From the Content Engine GUI, choose Caching > Customized Error Page. Use the displayed Custom Error Page Configuration window. For more information about how to use the Custom Error Page Configuration window, click the HELP button in the window.
- To use the Content Engine CLI to configure HTTP custom error pages, use the `http custom-error-page` EXEC command.

    `http custom-error-page download message url | reset { all | message } | upload { ip-address | hostname } dirname filename message`

Table 5-4 describes the parameters for the `http custom-error-page` command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>download</td>
<td>Copies the custom error message file to the Content Engine from the specified URL. To change the text for a specific message, use this option to identify the message you want to change, and specify the URL that is the source for the custom message file. The custom message file can be up to 16 KB in size and is used instead of the standard message page for the specified message.</td>
</tr>
<tr>
<td>url</td>
<td>Specifies the source of the custom error file. The file size cannot exceed 16 KB.</td>
</tr>
<tr>
<td>message</td>
<td>Specifies the type of custom error message (for example, ftp-put-error). See Table 5-3 for a list of these custom error messages.</td>
</tr>
<tr>
<td>reset</td>
<td>Reverts to the default error page.</td>
</tr>
<tr>
<td>message</td>
<td>Specifies the message that you want to return to the default page.</td>
</tr>
<tr>
<td>upload</td>
<td>Uploads the custom error message file to the specified host, directory, and file.</td>
</tr>
<tr>
<td>ip-address</td>
<td>Specifies the IP address of the host to which to copy the error page.</td>
</tr>
<tr>
<td>hostname</td>
<td>Specifies the hostname to which to copy the error page. The host should be reachable and allow copying a file to the specified directory.</td>
</tr>
<tr>
<td>dirname</td>
<td>Specifies the directory name to which to copy the error page.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the filename to which to copy the error page.</td>
</tr>
</tbody>
</table>

To display a list of the custom error messages:

```
ContentEngine# http custom-error-page download ?
```

To display a list of all the configured custom error pages:

```
ContentEngine# show http custom-error-page configured
```
To display the contents of the specified custom error page:

ContentEngine# show http custom-error-page custom-error page-filename

To copy a custom error message page to the Content Engine for the cache-read-error message:

ContentEngine# http custom-error-page download http://www.myserver.com/errors/cache-read-error.txt cache-read-error

To copy the current contents of the cache-read-error message to a file in the errors directory on the host with the IP address 192.168.1.1:

ContentEngine# http custom-error-page upload 192.168.1.1 /errors cache-read-error.txt cache-read-error

When the authentication method is NTLM, the Content Engine sends the proxy-allow-domain-error/www-allow-domain-error message to the end user if the user-supplied domain does not match the configured domains. You can customize the error that is sent to the end user as follows:

ContentEngine# http custom-error-page download proxy-allow-domain-error ?
WORD URL from where the file will be fetched, File size cannot exceed 16K

With certain browsers (for example, with the Netscape browser) the end user must specify the domain if the Content Engine does not have an NTLM default domain configured. If there is no default domain available for authentication, the proxy-no-default-domain-error/www-no-default-domain-error message is sent to the end user. You can customize the error message that is sent to the end user as follows:

ContentEngine# http custom-error-page download proxy-no-default-domain-error ?
WORD URL from where the file will be fetched, File size cannot exceed 16K

ContentEngine# http custom-error-page download www-no-default-domain-error ?
WORD URL from where the file will be fetched, File size cannot exceed 16K

To reset the cache-read-error message to the default text:

ContentEngine# http custom-error-page reset cache-read-error

To reset all of the configured custom error pages to the default text:

ContentEngine# http custom-error-page reset all

Creating a Custom Error-Signature Message for HTTP Custom Error Pages

In the ACNS 5.3.1 software and later releases, you can include an error signature on the HTTP custom error pages. To support this feature, the error-signature message identifier was added to the list of custom error messages (listed in Table 5-3).

The following is the default error signature that will be appended to the final error message page sent to the end user:

```html
<br clear="all">
<hr noshade size=1>
Generated %t by (a href="http://www.cisco.com/">Application and Content Networking System Software 5.3.0</a>)
</BODY></HTML>
```
If desired, you can create a custom error signature that will be appended to the final error message page instead of the default error signature. The following example shows an example of a custom error signature:

The request %R, from the Client %i does not conform to the HTTP Request. Please refer RFC-XXX, for further information on request format.
</body>
</html>

After creating a custom error signature, you use the `http custom-error-page download error-signature` EXEC command to copy the file that contains the custom error signature to the Content Engine. For example:

```
ContentEngine# http custom-error-page download error-signature ftp://somewhere/sig.htm
```

A custom error signature message supports dynamic data. This dynamic data is specific to the particular request and response that is to be shown with the error messages. Table 5-5 lists the argument specifiers for the error signature. These argument specifiers are replaced with the corresponding value when the error signature is appended to the final error message page that is sent to the end user.

<table>
<thead>
<tr>
<th>Argument Specifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Squid error code</td>
</tr>
<tr>
<td>d</td>
<td>Seconds elapsed since the request was received</td>
</tr>
<tr>
<td>e</td>
<td>Error number</td>
</tr>
<tr>
<td>E</td>
<td>Error string</td>
</tr>
<tr>
<td>h</td>
<td>Cache hostname (hostname of the Content Engine)</td>
</tr>
<tr>
<td>H</td>
<td>Server hostname</td>
</tr>
<tr>
<td>i</td>
<td>Client IP address</td>
</tr>
<tr>
<td>I</td>
<td>Server IP address</td>
</tr>
<tr>
<td>M</td>
<td>Request method</td>
</tr>
<tr>
<td>p</td>
<td>URL port # (number)</td>
</tr>
<tr>
<td>P</td>
<td>Protocol</td>
</tr>
<tr>
<td>R</td>
<td>Full HTTP request</td>
</tr>
<tr>
<td>t</td>
<td>Local time</td>
</tr>
<tr>
<td>T</td>
<td>Coordinated Universal Time (UTC)</td>
</tr>
<tr>
<td>U</td>
<td>URL without password</td>
</tr>
</tbody>
</table>

The following example shows how the template will be transformed into the final error message page that is sent to the end user. The template contains the following content for a request malformed error:

```
<html>
<body>
<h1> Malformed request </h1>
The request %R is not based on the Standard
</body>
</html>
```
You create a custom error signature that uses the “R” and “i” argument specifiers:

```html
<html>
<body>
<h1> Malformed request </h1>
The request %R , from the Client %i does not conform to the HTTP Request. Please refer RFC-XXX, for further information on request format.
</body>
</html>
```

When a request malformed error occurs for the client with the IP address of 192.168.192.161, the following custom error signature is appended to the final error message page that is sent to the end user (client 192.168.192.161):

```html
<html>
<body>
<h1> Malformed request </h1>
The request  GET http://www.abccorp.com HTTP/10, from the client 192.168.192.161 does not conform to the HTTP Request. Please refer RFC-XXX, for further information on request format.
</body>
</html>
```

Because the “R” and “i” argument specifiers were specified as part of the custom error signature, the actual full HTTP request (http://www.abccorp.com HTTP/10) and the IP address of the client (192.168.192.161) are included in the error signature of the final error message page that is sent to the end user.

### Creating Custom Messages for FTP Proxy Responses for FTP Native Requests

In the ACNS 5.4.1 software and later releases, you can use the `ftp-native custom-message EXEC` command to create, upload, and download files that contain the following custom messages:

- A custom FTP proxy-mode welcome message. The message is a custom welcome message that the Content Engine displays to the FTP client to welcome the FTP proxy-mode connection from the client. FTP clients include Reflection X clients, WS-FTP clients, or UNIX or DOS command line FTP programs, which are requesting a proxy-mode connection with the Content Engine in order to send nontransparent native FTP requests directly to the Content Engine.

- A custom error message used for when an FTP client is denied access based on the IP access control lists (ACLs) that have been defined for the native FTP proxy service.

If you intend to enable proxy authentication, then the welcome message should inform the user to authenticate with the proxy before logging in to the origin server, as shown in the following example:

Welcome to ce-Boxman. BigCorp’s Content Engine for Native FTP Proxy. Please login to the proxy with your username and password.

Alternatively, if you do not intend to enable proxy authentication, then the welcome message should inform the user to log in to the origin server using either the USER or SITE method, as shown in the following examples:

Welcome to ce-Boxman. BigCorp’s Content Engine for Native FTP Proxy. Please login to the origin server using the 'username@server-hostname' format.

Welcome to ce-Boxman. BigCorp’s Content Engine for Native FTP Proxy. Please login to the origin server using the 'SITE server-hostname' command followed by the 'USER username' command.
To use the Content Engine CLI to configure FTP custom message pages, use the `ftp-native custom-message` EXEC command.

```
ContentEngine# ftp-native custom-message ?
download   Download the custom message file specified by the URL to the CE
reset       Revert to default message and delete the local file on the CE
upload      Upload the custom message file to the specified host, directory and filename using the FTP protocol
```

```
ftp-native custom-message download { welcome welcome-message url | acl-denied acl-denied-message url } | reset { acl-denied | welcome | all } | upload { ip-address | hostname } dirname filename message
```

Table 5-6 describes the parameters for the `ftp-native custom-message` command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>download</td>
<td>Copies the custom message file to the Content Engine from the specified URL. To change the text for a specific message, use this option to identify the message you want to change, and specify the URL that is the source for the custom message file. The custom message file can be up to 16 KB in size and is used instead of the standard message for the specified message.</td>
</tr>
<tr>
<td>welcome</td>
<td>Indicates that you want to download the custom welcome message for the FTP proxy-mode welcome message.</td>
</tr>
<tr>
<td>acl-denied</td>
<td>Indicates that you want to download the custom error message that the Content Engine is to display to the FTP client because the client is being denied access based on the IP ACLs that have been defined for the native FTP proxy service.</td>
</tr>
<tr>
<td>message url</td>
<td>Specifies the URL from which the custom message file (the file that contains the FTP proxy-mode welcome message or the ACL access-denied error message) should be retrieved. The file size cannot exceed 16 KB.</td>
</tr>
<tr>
<td>reset</td>
<td>Specifies that the Content Engine (the FTP proxy) is to revert to the default message. Also deletes the local message files on the Content Engine.</td>
</tr>
<tr>
<td>welcome</td>
<td>Specifies that the Content Engine is to revert to the default FTP proxy-mode welcome message.</td>
</tr>
<tr>
<td>acl-denied</td>
<td>Specifies that the Content Engine is to revert to the default ACL access-denied error message.</td>
</tr>
<tr>
<td>all</td>
<td>Specifies that the Content Engine is to revert to all default FTP proxy messages.</td>
</tr>
<tr>
<td>upload</td>
<td>Uploads the custom message file from the Content Engine to the specified host, directory, and file.</td>
</tr>
<tr>
<td>ip-address</td>
<td>Specifies the IP address of the host to which the Content Engine is to copy the custom message file.</td>
</tr>
<tr>
<td>hostname</td>
<td>Specifies the hostname to which the Content Engine is to copy the file that contains the custom message. The host should be reachable and allow a file to be copied to the specified directory.</td>
</tr>
<tr>
<td>dirname</td>
<td>Specifies the directory name to which to copy the custom message file.</td>
</tr>
<tr>
<td>filename</td>
<td>Specifies the filename to which to copy the custom message file.</td>
</tr>
</tbody>
</table>
To display a list of the names of the configured FTP native custom messages:

```
ContentEngine# show ftp-native custom-message
```

To display the contents of the local copy of the specified custom message (for example, the ACL-denied message or the welcome message that has been downloaded to the Content Engine) on the CLI display screen:

```
ContentEngine# show ftp-native custom-message {acl-denied acl-denied-filename | welcome welcome-filename}
```

To copy the FTP native custom welcome message to the Content Engine:

```
ContentEngine# ftp-native custom-message download welcome http://www.myserver.com/errors/ftp-native-welcome.txt
```

## Shutting Down Standalone Content Engines

A controlled shutdown refers to the process of properly shutting down a Content Engine without turning off the power on the device. With a controlled shutdown, all of the application activities and the operating system are properly stopped on a standalone Content Engine but the power is still on. Controlled shutdowns of a standalone Content Engine can help you minimize the downtime when the Content Engine is being serviced.

### Caution

If a controlled shutdown is not performed, the Content Engine file system can be corrupted. It also takes longer to reboot the Content Engine if the Content Engine is not properly shut down.

The `shutdown` EXEC command enables you to shut down and optionally power off a Content Engine.

- **Shut down** means that all application activities (applications and operating system) are stopped, but the power is still on. This is a shutdown only, and is similar to the Linux `halt` command.
- **Shut down power off** means that the Content Engine is powered down by the ACNS software after being shut down. This operation is also referred to as a `software poweroff`. The implementation of the shutdown poweroff feature uses the advanced configuration and power interface (ACPI) power management interface.

When a shutdown poweroff is performed on a standalone Content Engine, these conditions result:

- All application activities are stopped on the Content Engine, and the Content Engine is shut down through the ACNS software.
- Power is turned off through a software power off.
- The fan is not running; however, the power LED is flashing on the Content Engine.

### Note

Content Engines cannot be powered on again through software after a software poweroff. You must press the power button once on these Content Engines to bring these Content Engines back online.
Table 5-7 describes the shutdown-only operation and the shutdown poweroff operation for standalone Content Engines.

### Table 5-7 Shutting Down Standalone Content Engines Through CLI Commands

<table>
<thead>
<tr>
<th>Activity</th>
<th>All Content Engine Models Supported in the ACNS 5.3.1 Software or Later</th>
<th>Content Engines with Power Management Capability in ACNS 5.3.1 Software or Later</th>
</tr>
</thead>
<tbody>
<tr>
<td>User performs a shutdown operation on the Content Engine</td>
<td>Shutdown only</td>
<td>Shutdown poweroff</td>
</tr>
<tr>
<td></td>
<td>ContentEngine# shutdown</td>
<td>ContentEngine# shutdown poweroff</td>
</tr>
<tr>
<td>User intervention to bring Content Engine back online</td>
<td>To bring a Content Engine that has an on/off switch on the back (for example, the CE-507, CE-507AV, CE-560, CE-560AV, or the CE-590) back online after a shutdown operation, turn the on/off switch twice. To bring a Content Engine that has a power button (instead of an on/off switch on the back) back online after a shutdown operation: first press and hold the power button for several seconds to power off these models, and then press the power button once again.</td>
<td>After a shutdown poweroff, you must press the power button once to bring the Content Engine back online.</td>
</tr>
<tr>
<td>File system check</td>
<td>Will not be performed after you turn the power on again and reboot the Content Engine.</td>
<td>Will not be performed after you turn the power on again and reboot the Content Engine.</td>
</tr>
</tbody>
</table>

All of the Content Engine models that are supported in the ACNS 5.3 release support the shutdown feature (the `shutdown EXEC` command), which was added in the ACNS 5.3.1 software release. However, the shutdown poweroff feature (the `shutdown poweroff EXEC` command) is only supported on the newer Content Engine models that support the power management capability (for example, the CE-510, CE-510A, CE-511, CE-511A, CE-565, CE-565A, CE-566, CE-566A, CE-7305, CE-7305A, CE-7320, CE-7325, and CE-7326).

The `shutdown EXEC` command is supported in all device modes (Content Distribution Manager, Content Engine, Content Router, and IP/TV Program Manager). For a description of how to use the `shutdown EXEC` command with a standalone Content Engine (a Content Engine that is not registered with a Content Distribution Manager), see the next section, “Shutting Down Standalone Content Engines from the Command Line.” For information about how to use the `shutdown EXEC` command with other types of devices (for example, a Content Distribution Manager, a Content Engine that is registered with a Content Distribution Manager, or a Content Router), see the *Cisco ACNS Software Command Reference, Release 5.5* publication.

### Shutting Down Standalone Content Engines from the Command Line

You can enter the `shutdown EXEC` command from a console session or from a remote session (Telnet or SSH Version 1 or Version 2) to shut down a standalone Content Engine.

To perform a shutdown on a standalone Content Engine, follow these steps:

**Step 1** Enter the `shutdown EXEC` command.

```
ContentEngine# shutdown
```
Step 2  When you are asked if you want to save the system configuration, enter yes.
System configuration has been modified. Save?[yes]:yes

Step 3  When you are asked if you want to proceed with the shutdown, press Enter to proceed with the shutdown operation.
Device can not be powered on again through software after shutdown.
Proceed with shutdown?[confirm]

A message appears, reporting that all services are being shut down on this Content Engine.
Shutting down all services, will timeout in 15 minutes.
shutdown in progress ..System halted.

Step 4  After the system is shut down (the system has halted), an ACNS software shutdown shell displays the current state of the system (for example, “System has been shut down”) on the console. You are asked whether you want to perform a software power off (the Power down system by software option), or if you want to reload the system through the software.

================= SHUTDOWN SHELL =================
System has been shut down.
You can either
Power down system by pressing and holding power button
or
1. Reload system through software
2. Power down system through software

Step 5  To power down the Content Engine, press and hold the power button on the Content Engine, or use one of the following methods to perform a shutdown poweroff:

- From the console command line, enter 2 when prompted, as follows:

  ================== SHUTDOWN SHELL =================
  System has been shut down.
  You can either
  Power down system by pressing and holding power button
  or
  1. Reload system through software
  2. Power down system through software

- From the Content Engine CLI, follow these steps:
  a. Enter the shutdown poweroff EXEC command.

  ContentEngine# shutdown poweroff

  b. When you are asked if you want to save the system configuration, enter yes.

  System configuration has been modified. Save?[yes]:yes

  c. When you are asked to confirm your decision, press Enter.

  Device can not be powered on again through software after poweroff.
  Proceed with poweroff?[confirm]
  Shutting down all services, will timeout in 15 minutes.
  poweroff in progress ..Power down.
Removing or Replacing a Content Engine

Shutting Down and Rebooting Standalone Content Engines from the Content Engine GUI

You can use the Content Engine GUI to perform a controlled shutdown on a standalone Content Engine. The Content Engine performs a controlled shutdown and then restarts the operating system on the Content Engine.

The Content Engine releases all WCCP connections to a router during the reboot process if these conditions exist:

- The Clean WCCP shutdown check box is checked in the main window of the Content Engine GUI.
- WCCP Version 2 is enabled on the Content Engine.

To use the Content Engine GUI to perform a controlled shutdown on a standalone Content Engine, follow these steps:

**Step 1**
From the Content Engine GUI, click the **Reboot** button in the Content Engine main window (Figure 4-17).

The Content Engine performs the controlled shutdown and then restarts the operating system on the Content Engine.

**Step 2**
To reboot this standalone Content Engine, click the **Reboot** button in the Content Engine main window.

Tip
If the Content Engine main window (Figure 4-17) is not currently displayed in your browser, click the words “Content Engine” in the upper-left corner any Content Engine GUI window to return to the Content Engine main window.

**Step 3**
When you are prompted to confirm your decision, click **OK** to begin rebooting this standalone Content Engine.

Removing or Replacing a Content Engine

See 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 Content Engine is no longer responding to it, the router stops sending requests to the Content Engine. This is transparent to users. If other Content Engines are attached to the router, the router continues sending requests to the other Content Engines.

Remotely Upgrading the BIOS

In the ACNS 5.3.1 software and later releases, you can perform a BIOS upgrade remotely through the CLI. This feature is currently only supported for the Content Engine model CE-7326. For more information on this topic, see the *Cisco ACNS Software Upgrade and Maintenance Guide, Release 5.x*. 
Configuring Transparent Redirection for Standalone Content Engines

This chapter discusses the following methods for transparently redirecting content requests to standalone Content Engines:

- Web Cache Communication Protocol (WCCP)-enabled routers that intercept content requests and redirect them to standalone Content Engines
- A Layer 4 switch that intercepts content requests and redirects them to standalone Content Engines

This chapter includes the following sections:

- Overview of WCCP Transparent Redirection, page 6-2
- Configuring Standalone Content Engines for WCCP Transparent Redirection, page 6-9
- Disabling and Reenabling WCCP Flow Redirection on Standalone Content Engines, page 6-17
- Shutting Down WCCP on Standalone Content Engines, page 6-17
- Configuring a Router for WCCP Transparent Redirection, page 6-18
- Configuring WCCP Services on a Router, page 6-27
- Clearing WCCP Statistics on a Router, page 6-34
- Configuring WCCP Layer 2 Support, page 6-34
- Examples of Configuring WCCP Services for Standalone Content Engines, page 6-38
- Configuring Layer 4 Switching as a Redirection Method, page 6-50

The Cisco ACNS transparent caching solution uses a WCCP-enabled router and various advanced techniques to ensure that the Content Engine remains transparent, even if web browsers are nonoperational or web servers are not HTTP-compliant. One of these techniques is the bypass feature. For information about how to configure the bypass feature, see the “Configuring Bypass Settings on Standalone Content Engines” section on page 15-3.

For complete syntax and usage information for the Content Engine CLI commands used in this chapter, refer to the Cisco ACNS Software Command Reference, Release 5.5 publication.

For further information on WCCP, refer to the Cisco IOS Configuration Fundamentals Configuration Guide and the Cisco IOS Configuration Fundamentals Command Reference.
Overview of WCCP Transparent Redirection

Cisco developed WCCP within Cisco IOS software to enable routers or switches to intercept packets based on the IP, UDP, and TCP header information and then redirect those packets transparently to Content Engines running the ACNS 5.x software.

There are two versions of WCCP, Version 1 and Version 2. WCCP Version 1 only supports one WCCP service (the standard web-cache service [service 0]) and a single router. The main features of WCCP Version 1 include:

- Support for only one router (home router)
- Support for traffic redirection on port 80 only
- Support for up to 32 Content Engines per WCCP service
- No bypass support (for example, static bypass, error bypass, and authentication bypass are not supported)
- No generic routing encapsulation (GRE) on return

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, RealAudio, and video.

We recommend that you use WCCP Version 2 because it supports a wider set of features and services (see Table 6-3) as well as multiple routers.

Note: WCCP works with IP networks only.

Redirection of Packets with WCCP

When transparent redirection with a WCCP-enabled router is used to redirect requests to a Content Engine, the web clients send their content requests to the source and are not aware that their requests are being redirected to the Content Engine by a WCCP-enabled router. Because this interception and redirection process is completely invisible, or transparent, to the client who is requesting the content, no desktop changes are required (clients do not have to configure their browsers or media players to point to a specific proxy server). The Content Engine operation is transparent to the network; the WCCP-enabled router operates entirely in its normal role for nonredirected traffic.

Figure 6-1 shows the packet flow between a Content Engine and a router.
In Figure 6-1, if the Content Engine does not have the data, a cache miss occurs, and the Content Engine sends the request to the origin server. As the Content Engine receives the data from the origin server, it saves the data and forwards the data to the client.

To use WCCP transparent redirection, you must first define a WCCP service on the WCCP-enabled router. The parameters for a given service are its name, service identifier (service number), and the router interface that is to be used to support this WCCP service.

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.

For information about configuring healing mode for Content Engines that are part of a Content Engine cluster, see the “Configuring Healing Mode for Content Engine Clusters” section on page 7-70. For information about configuring advanced caching features such as WCCP slow start, see Chapter 15, “Configuring Advanced Transparent Caching Features on Standalone Content Engines.”
Overview of WCCP Transparent Redirection

Redirection of Packets with Content Engine Clusters and WCCP-Enabled Routers

If there is a cluster (group) of Content Engines, the one seen by all the WCCP Version 2-enabled routers and the one that has the lowest IP address becomes the lead Content Engine. The role of this lead Content Engine is to determine how traffic should be allocated across the Content Engines in the cluster. The assignment information is passed to the entire service group from the designated lead Content Engine so that the WCCP-enabled routers of the group can redirect the packets properly and the Content Engines in the group can better manage their load.

The following describes how a Content Engine in a Content Engine cluster is designated the lead:

1. Each Content Engine is configured with a list of WCCP-enabled routers.

   With WCCP Version 1, only a single WCCP-enabled router services a cluster, becoming the default home router for the cluster.

   With WCCP Version 2, multiple WCCP-enabled routers (each router list can contain up to eight 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.

2. Each Content Engine announces its presence to each router on the router list. The routers reply with their view of Content Engines in the service group.

3. After the view is consistent across all of the Content Engines in the cluster, one Content Engine is designated the lead Content Engine and sets the policy that the WCCP-enabled routers need to deploy in redirecting packets.

About Dynamic Load Distribution with WCCP Version 2

When an IP packet is received by a WCCP Version 2-enabled router, it is examined to determine if it is a request that should be directed to a Content Engine. This is done by matching the request to the defined service criteria. These packets are passed to the router’s processing routine to determine which Content Engine, if any, should receive the redirected packets.

The determination of which Content Engine should receive the intercepted packets is made by performing a hash function on the destination IP address to obtain an address bucket to which the packet is assigned. These address buckets are then mapped to a particular Content Engine depending on how many Content Engines are present and how busy they are. See Figure 6-2.
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.

WCCP Version 2 supports dynamic load distribution that allows the routers to adjust the loads being forwarded to the individual Content Engines in the cluster. WCCP uses two techniques to perform this task:

- Load balancing allows the set of hash address 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 WCCP-enabled router to selectively redirect the load to avoid exceeding the capacity of the Content Engines.

### About Packet-Forwarding Methods

A WCCP-enabled router redirects intercepted requests to a standalone Content Engine using two packet-forwarding methods:

- Generic routing encapsulation (GRE)—Allows packets to reach the Content Engine even if there are any number of routers in the path to the Content Engine.
- Layer 2 redirection—Allows packets to be switched at Layer 2 (MAC layer) and reach the Content Engine.

Table 6-1 describes these two methods.
In both packet-forwarding methods, the hash parameters specify how redirected traffic should be load balanced among the Content Engines in the various WCCP service groups.

The term assignment method denotes the method used by WCCP to perform load distribution across Content Engines. There are two possible load-balancing assignment methods: hashing and masking. If the mask load-balancing method is not specified, then the hash load-balancing method (see Figure 6-2), which is the default method, is used.

The redirection mode is controlled by the Content Engine. The first Content Engine that joins the WCCP service group decides the forwarding method (GRE or Layer 2 redirection) and the assignment method (hashing or masking). The term mask assignment is used to refer to WCCP Layer 2 Policy Feature Card 2 (PFC2) input redirection.

Regarding fallback capabilities for the forwarding method, if the router is using the GRE redirection because other Content Engines are using GRE for the web-cache service and the new Content Engine has been configured to use Layer 2 redirection (the wccp web-cache router-list-num 1 l2-redirect global configuration command has been specified), the Content Engine attempts to negotiate Layer 2 redirection. The Content Engine will advertise its Layer 2 redirection capability but when the router replies with GRE, the Content Engine will fall back to GRE. This is expected behavior because it is currently not possible to configure the Content Engine to only use Layer 2 redirection.

Regarding fallback capabilities for the assignment method, the Content Engine falls back to the assignment method supported in hardware unless the assign-method-strict option is used (for example, if the wccp https-cache assign-method-strict command is used to specify the assign-method-strict option for the https-cache service) rather than remain out of the Content Engine cluster indefinitely. If masking is selected with WCCP output redirection, then the Content Engine falls back to the original hardware acceleration that is used with the Multilayer Switch Feature Card (MSFC) and the Policy Feature Card (PFC).

For example, 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.

<table>
<thead>
<tr>
<th>Packet-Forwarding Method</th>
<th>Load-Balancing Method: Hashing</th>
<th>Load-Balancing Method: Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRE (Layer 3)</td>
<td>Packet redirection is completely handled by the router software.</td>
<td>Packet redirection is handled by the router software. Mask assignment is not recommended when GRE is being used as the packet-forwarding method.</td>
</tr>
<tr>
<td>Layer 2 redirection</td>
<td>First redirected packet is handled by the router software; all subsequent redirected packets are handled by the router hardware.</td>
<td>All packets are handled by the router hardware (currently supported on only the Catalyst 6000 series switches because special hardware is required).</td>
</tr>
</tbody>
</table>
The following list includes some typical reasons why a Content Engine would reject packets and initiate packet return:

- 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 a server error or authentication failure. In this situation, the client can reach the server directly. The Content Engine, therefore, is not the reason for the failure.
- The Content Engine is filtering out certain conditions that make processing packets counterproductive, for example, when IP authentication has been turned on.
- You have configured a static bypass list on the Content Engine.

**Note**

The packets are redirected to the source of the connection between the WCCP-enabled router and the Content Engine. Depending on the Cisco IOS software 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 IP address of the WCCP-enabled router stored in the router list. For more information on router lists, see the “Defining Router Lists on Standalone Content Engines” section on page 6-12.

Cisco Express Forwarding (CEF) has been integrated into WCCP Version 2 to achieve optimal performance during packet redirection. WCCP Version 2 also allows you to configure multiple routers (router lists) to support a particular WCCP service (for example, RTSP redirection), as described in the “Configuring Standalone Content Engines for WCCP Transparent Redirection” section on page 6-9. For a list of supported WCCP Version 2 features and services, see Table 6-3.

For more information about these packet-forwarding methods, see:

- Using Layer 3 GRE as a Packet-Forwarding Method, page 6-7
- Using Layer 2 Redirection as a Packet-Forwarding Method, page 6-8

**Using Layer 3 GRE as a Packet-Forwarding Method**

GRE is a Layer 3 technique that allows datagrams to be encapsulated into IP packets at the WCCP-enabled router and then redirected to a Content Engine (the transparent proxy server). At this intermediate destination, the datagrams are decapsulated and then routed to an origin server to satisfy the request if a cache miss occurs. In doing so, the trip to the origin server appears to the inner datagrams as one hop. Usually, the redirected traffic using GRE is referred to as GRE tunnel traffic. With GRE, all redirection is handled by router software.

With WCCP redirection, a Cisco router does not forward the TCP SYN packet to the destination because the router has WCCP enabled on the destination port of the connection. Instead, the WCCP-enabled router encapsulates the packet using GRE tunneling and sends it to the standalone Content Engine that has been configured to accept redirected packets from this WCCP-enabled router.
After receiving the redirected packet, the Content Engine does the following:

1. Strips the GRE layer from the packet.
2. Decides whether it should accept this redirected packet and process the request for content.
   a. If the Content Engine decides to accept the request, it sends a TCP SYN ACK packet to the client. In this response packet, the Content Engine uses the IP address of the original destination (origin server) that was specified as the source address. This is done so that the Content Engine can be invisible (transparent) to the client; it pretends to be the destination that the client’s TCP SYN packet was trying to reach.
   b. If the Content Engine decides not to accept the request, it reencapsulates the TCP SYN packet in GRE, and sends it back to the WCCP-enabled router. The router understands that in this case the Content Engine is not interested in this connection, and forwards the packet to its original destination (that is, the origin server).

For example, a standalone Content Engine would decide not to accept the request because it is configured to bypass requests that originate from a certain set of clients or that are destined to a particular set of servers.

### Using Layer 2 Redirection as a Packet-Forwarding Method

Layer 2 redirection is a term for a situation 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 router hardware at Layer 2. This type of redirection is currently supported only with the Cisco Catalyst 6000 and 6500 series switches. With Layer 2 redirection, the first redirected traffic packet is handled by the router software. The rest of the traffic is handled by the router hardware. With Layer 2 redirection, the Content Engine instructs the router or switch to apply a bitmask to certain packet fields, which in turn provides a mask result or index mapped to the Content Engines in the cluster in the form of a mask index address table. The redirection process is accelerated in the switching hardware, which makes Layer 2 redirection more efficient than Layer 3 GRE.

**Note**

WCCP is only licensed on the Content Engine and not on the redirecting router. WCCP does not interfere with normal router or switch operations.

For information about configuring Layer 2 redirection, see the “Configuring WCCP Layer 2 Support” section on page 6-34.
Configuring Standalone Content Engines for WCCP Transparent Redirection

The type of WCCP services supported by a standalone Content Engine and a WCCP-enabled router varies based on whether WCCP Version 1 or Version 2 is used, as indicated in Table B-3. All of the services except for the standard web-cache service (service 0) requires that WCCP Version 2 (as opposed to WCCP Version 1) is running on the router and the standalone Content Engine for a particular WCCP service to be supported. Table 6-2 lists the supported WCCP transparent redirection services for standalone Content Engines.

Table 6-2 Supported WCCP Transparent Redirection Services

<table>
<thead>
<tr>
<th>WCCP Transparent Redirection Service</th>
<th>WCCP Service Name</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service 80</td>
<td>rtsp</td>
<td>RTSP redirection service used to redirect RTSP requests from RealMedia clients to the Content Engine. For transparent redirection of RealMedia requests, you only need to configure service 80. For WMT RTSP transparent redirection support (transparent redirection of RTSP requests from Windows Media 9 players), you must configure service 80 as well as service 83.</td>
<td>When the clients are RealMedia players, the backend RTSP server is the RealProxy server. When the clients are Windows Media 9 players, the backend RTSP server is the Windows Media 9 server that is running on the Content Engine.</td>
</tr>
<tr>
<td>Service 81</td>
<td>mmstu</td>
<td>MMSTU redirection service used to redirect MMS-over-HTTP requests from WMT clients to the Content Engine.</td>
<td>Clients are WMT clients (for example, Windows Media players Version 6, 7, or 9).</td>
</tr>
<tr>
<td>Service 82</td>
<td>mmsu</td>
<td>MMSU redirection service used to redirect MMS-over-HTTP requests from WMT clients to the Content Engine. For MMS redirection of WMT requests, you must configure service 81 and 82 on the router.</td>
<td>Clients are WMT clients (for example, Windows Media players Version 6, 7, or 9).</td>
</tr>
<tr>
<td>Service 83</td>
<td>wmt-rtspu</td>
<td>WMT RTSP redirection service used to redirect RTSP requests from WMT clients (Windows Media 9 Players) to the Content Engine (that is functioning as a Windows Media 9 server).</td>
<td>Clients and server are WMS 9 (for example, Windows Media 9 players and the Windows Media Series 9 server that is running on the Content Engine and functioning as the backend RTSP server for WMT RTSP requests).</td>
</tr>
</tbody>
</table>
To view a list of WCCP options and services that you can configure on a standalone Content Engine, enter the `wccp` EXEC command followed by a question mark (“?”). The following sample output is from a Content Engine that has WCCP Version 2 enabled. If you are using WCCP Version 1, only one router is supported (home router) and a single WCCP service (the standard web-cache service).

```
ContentEngine(config)# wccp ?
access-list Configure an IP access-list for inbound WCCP encapsulated traffic
custom-web-cache Custom web caching service
dns Caching Domain Name Service
flow-redirect Redirect moved flows
ftp-native Transparent FTP proxy caching service
home-router WCCP Version 1 Home Router Ip address
https-cache HTTPS caching service
port-list Port list for use in WCCP service
reverse-proxy Reverse Proxy web caching service
router-list Router List for use in WCCP services
rtsp RTSP protocol transparent interception
service-number WCCPv2 service number
shutdown Wccp Shutdown parameters
slow-start accept load in slow-start mode
spoof-client-ip Use client IP while connecting to the origin server
version WCCP Version Number
web-cache Standard web caching service
wmt Windows media caching service
wmt-rtspu Windows media RTSPU (port 5005) transparent interception
```

For information about configuring the standard web-cache service (service 0) using WCCP Version 1, see the “Example 1—Configuring the Web-Cache Service with WCCP Version 1” section on page 6-39.

See Table 6-3 for a list of these WCCP Version 2 options and services. An asterisk (“*”) in Table 6-3 indicates whether a particular capability is a WCCP option or a WCCP service.

### Table 6-3  WCCP Version 2 Options and Services

<table>
<thead>
<tr>
<th>WCCP Option or Service Name</th>
<th>WCCP Option</th>
<th>WCCP Service</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-list</td>
<td>*</td>
<td></td>
<td>Configuring WCCP Access Lists for Standalone Content Engines</td>
</tr>
<tr>
<td>custom-web-cache</td>
<td>*</td>
<td></td>
<td>Configuring the Custom-Web-Cache Service (Service 98) on a Router</td>
</tr>
<tr>
<td>dns</td>
<td>*</td>
<td></td>
<td>Configuring the DNS Caching Service (Service 53) on a Router</td>
</tr>
<tr>
<td>flow-redirect</td>
<td>*</td>
<td></td>
<td>Disabling and Reenabling WCCP Flow Redirection on Standalone Content Engines</td>
</tr>
<tr>
<td>ftp-native</td>
<td>*</td>
<td></td>
<td>Configuring the FTP-Native Caching Service (Service 60) on a Router</td>
</tr>
<tr>
<td>https-cache</td>
<td>*</td>
<td></td>
<td>Configuring the HTTPS-Cache Service (Service 70) on a Router</td>
</tr>
<tr>
<td>port-list</td>
<td>*</td>
<td></td>
<td>Defining Port Lists on Standalone Content Engines</td>
</tr>
<tr>
<td>reverse-proxy</td>
<td>*</td>
<td></td>
<td>Configuring the Reverse-Proxy Service (Service 99) on a Router</td>
</tr>
<tr>
<td>rtsp</td>
<td>*</td>
<td></td>
<td>Configuring the RTSP Service (Service 80) on a Router</td>
</tr>
<tr>
<td>service-number</td>
<td>*</td>
<td></td>
<td>Configuring User-Defined WCCP Services (Services 90–97) on a Router</td>
</tr>
<tr>
<td>shutdown</td>
<td>*</td>
<td></td>
<td>Defining Port Lists on Standalone Content Engines</td>
</tr>
<tr>
<td>slow-start</td>
<td>*</td>
<td></td>
<td>Configuring WCCP Slow Start</td>
</tr>
<tr>
<td>spoof-client-ip</td>
<td>*</td>
<td></td>
<td>Configuring WCCP IP Spoofing</td>
</tr>
<tr>
<td>version</td>
<td>*</td>
<td></td>
<td>Enabling WCCP on Standalone Content Engines</td>
</tr>
</tbody>
</table>
### Configuring Standalone Content Engines for WCCP Transparent Redirection

#### Chapter 6      Configuring Transparent Redirection for Standalone Content Engines

#### Table 6-3  WCCP Version 2 Options and Services (continued)

<table>
<thead>
<tr>
<th>WCCP Option or Service Name</th>
<th>WCCP Option</th>
<th>WCCP Service</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-cache</td>
<td>*</td>
<td>Example 2—Configuring the Web-Cache Service with WCCP Version 2</td>
<td></td>
</tr>
<tr>
<td>wmt-rtspu</td>
<td>*</td>
<td>Configuring the WMT-RTSPU Service (Service 83) on a Router</td>
<td></td>
</tr>
</tbody>
</table>

#### Note

In the ACNS software releases earlier than Release 5.2, a maximum of eight active WCCP services were supported by a WCCP Version 2-enabled router and a Content Engine. The ACNS 5.2.1 software and later releases up to 25 active WCCP Version 2 services are supported. In the ACNS 5.3.1 software release, the wmt-rtspu (Service 83) was added. In the ACNS 5.3.1 software release, there are currently 18 services that can be configured.

In the ACNS 5.2 software, another interception mode (the accept-all mode) was added for the https-cache service. This mode was added to support the filtering of HTTPS traffic. This mode works the same way as traditional WCCP services (for example, the standard web-cache service [service 0] that intercepts all web traffic by default). If the `wccp https-cache accept-all` global configuration command is used, the HTTPS cache (the Content Engine that has the https-cache service configured and enabled) will work in accept all mode (all HTTPS traffic is intercepted by the Content Engine); otherwise, the HTTPS cache works in accept only mode, as in the ACNS 5.1.x software.

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

- The Content Engine must be configured to accept redirected packets from one or more WCCP-enabled routers. With WCCP Version 1, only a single router (home router) is supported. With WCCP Version 2, multiple routers (router lists) are supported.
- Versions of software on the Content Engines must be compatible with those installed on the WCCP-enabled router.
- The Content Engines must not have their packets encrypted or compressed and should be part of the inside Network Address Translation (NAT) firewall 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.
- After enabling WCCP on the router, you must configure the router and the Content Engine for transparent caching services.

For information about how to configure standalone Content Engines for WCCP transparent redirection, see:

- Enabling WCCP on Standalone Content Engines, page 6-12
- Defining Port Lists on Standalone Content Engines, page 6-12
- Defining Router Lists on Standalone Content Engines, page 6-12
- Configuring WCCP Services on Standalone Content Engines, page 6-14
- Displaying WCCP Configuration Information for Standalone Content Engines, page 6-16

For sample scenarios of how to configure WCCP services for standalone Content Engines, see the “Examples of Configuring WCCP Services for Standalone Content Engines” section on page 6-38. For information about how to configure WCCP transparent interception on a router, see the “Configuring a Router for WCCP Transparent Redirection” section on page 6-18.
Enabling WCCP on Standalone Content Engines

To enable WCCP on a standalone Content Engine, enter the `wccp version` global configuration command. Specify the WCCP version that you want the standalone Content Engine to run. Ensure that the routers used in the WCCP environment are running a software version that supports the WCCP version configured on the standalone Content Engine.

The following example shows how to enable WCCP Version 2 on a standalone Content Engine:

```
Content Engine (config)# wccp version 2
```

The following example shows how to enable WCCP Version 1 on a standalone Content Engine:

```
Content Engine (config)# wccp version 1
```

Only one version of WCCP can be enabled on a Content Engine at the same time. We recommend that you run WCCP Version 2 because it supports a broader set of WCCP options and services and provides multiple router support (router lists). See Table 6-3 for a list of the features and services supported by WCCP Version 2.

It is not necessary to disable WCCP Version 1 before enabling WCCP Version 2, and vice versa. However, to properly shut down WCCP on a Content Engine, you must disable the currently running version as described in the “Disabling and Reenabling WCCP Flow Redirection on Standalone Content Engines” section on page 6-17.

Defining Port Lists on Standalone Content Engines

With WCCP Version 1, web-cached information can only be redirected to a Content Engine 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, web caching for ports other than port 80, RealAudio, and video. If a router is configured for WCCP Version 2 instead of WCCP Version 1, then additional TCP ports other than port 80 can be configured on the WCCP-enabled router to redirect traffic to a Content Engine.

You can configure up to eight port lists (port lists number 1 through 8). These port lists specify the port numbers on which the Content Engine will listen for incoming WCCP redirected traffic. These ports lists allow you to configure the Content Engine to listen for incoming WCCP requests on more than one port.

By default, the Content Engine listens for incoming traffic on port 80. Create one port list for each of the eight user-defined WCCP services that you will be creating (services 90 to 97). You can define up to eight ports per port list. In this case, each port list has a single port (for example, port list 1 consists of port 32).

```
ContentEngine(config)# wccp port-list 1 32
ContentEngine(config)# wccp port-list 2 33
ContentEngine(config)# wccp port-list 3 34
ContentEngine(config)# wccp port-list 4 35
ContentEngine(config)# wccp port-list 5 36
ContentEngine(config)# wccp port-list 6 37
ContentEngine(config)# wccp port-list 7 38
ContentEngine(config)# wccp port-list 8 39
```

Defining Router Lists on Standalone Content Engines

As part of configuring a WCCP Version 2 service on a Content Engine, you must create a list of WCCP Version 2-enabled routers that will support a specific WCCP Version 2 service (for example, the rtsp service) for the Content Engine.
To create a router list on a standalone Content Engine, use the `wccp router-list` global configuration command. Enter the IP address of every WCCP Version 2-enabled router that will support a particular WCCP service for the Content Engine. If different routers will be used for different WCCP services, you must create more than one router list. Each router list can contain up to eight routers.

In the following example, router list number 1 is created, and it contains a single router (the WCCP Version 2-enabled router with IP address 10.10.10.1):

```
ContentEngine(config)# wccp router-list 1 10.10.10.1
```

The following example shows how to create a router list (router list 1) and then configure the Content Engine to accept redirected WMT traffic (the WCCP service named “wmt”) from the WCCP Version 2-enabled router on router list 1:

```
ContentEngine(config)# wccp router-list 1 10.10.10.2
ContentEngine(config)# wccp wmt router-list 1
ContentEngine(config)# wccp version 2
```

With WCCP Version 1, you can only configure a single WCCP-enabled router to support the standard web-cache service (service 0). Even if there is a cluster of Content Engines, only a single WCCP Version 1-enabled router services a cluster of Content Engines, becoming the default home router for the cluster. For information about how to configure a router as a home router for the standard web-cache service using WCCP Version 1, see the “Example 1—Configuring the Web-Cache Service with WCCP Version 1” section on page 6-39.

**Note**

The `ip wccp` global configuration command must be used to enable WCCP on each router that is included on the router list.

When configuring a Content Engine for WCCP Version 2, you can configure an IP multicast address instead of a list of routers on the Content Engine. The use of a list of routers on the Content Engine precludes the need to use IP multicast but requires more configuration on each Content Engine. The use of an IP multicast address reduces the configuration on the Content Engine as well as the protocol overhead.

With IP multicasting, an IP multicast address is configured on the Content Engine. The WCCP Version 2-enabled routers are configured to receive the IP multicast address on one or more interfaces. These routers then send their redirected requests to the specified IP multicast address on the Content Engine. Multicast addresses must be between 224.0.0.0 and 239.255.255.255. The Internet Assigned Numbers Authority (IANA) controls the assignment of IP multicast addresses. The IANA has assigned the IPv4 Class D address space to be used for IP multicast. Therefore, all IP multicast group addresses fall in the range from 224.0.0.0 through 239.255.255.255. However, some combinations of source and group address should not be routed for multicasting purposes. For a list of the unusable multicast address ranges and the reasons they should not be used, see the “Unusable Multicast Address Assignments” section on page B-11.

The additional configuration required on the WCCP Version 2-enabled routers that are intended to become members of the service group when IP multicast is used is as follows:

- The IP multicast address for use by the service group must be configured.
- The interface or interfaces that the WCCP Version 2-enabled router wants to receive the IP multicast address need to be configured with the `ip wccp {web-cache | service-number} group-listen` command.

For network configurations in which another router must be traversed to get to the target router, the router being traversed must be configured to perform IP multicast routing:
IP multicast routing needs to be enabled by configuring it for the router with the `ip multicast-routing` command.

The router interfaces that connect to the Content Engines must be configured to receive multicast with an `ip pim` command.

## Configuring WCCP Services on Standalone Content Engines

Some of the WCCP services that WCCP-enabled routers and Content Engines can support have a well-known set of criteria and a predefined service identifier (for example, the standard web-cache service [service 0]). These services are called *predefined* WCCP services. Other examples include the reverse-proxy caching service (service 99), the https-caching service (service 70), and the rtsp service (service 80).

Other WCCP services that are not well-known may be defined by specifying a set of criteria and assigning these user-defined WCCP Version 2 services a service identifier. WCCP Version 2 allows you to define up to eight user-defined WCCP services (services 90 to 97). Each of these user-defined services supports up to eight ports.

For more information about configuring a standalone Content Engine to support a user-defined WCCP service, see the “Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 6-15.

*Note*

See Table B-3 for a list of supported WCCP services.

Keep these important points in mind when configuring a WCCP services for standalone Content Engines:

- WCCP Version 1 only supports one WCCP service (the standard web-cache service [service 0]) and a single router. Consequently, we recommend that you use WCCP Version 2 because it supports a wider set of features and services as well as multiple routers.
- You must create a list of WCCP Version 2-enabled routers that will support a specific WCCP service used on the standalone Content Engine. If different routers will be used for different WCCP services, it is necessary to create more than one router list.
- WCCP Version 2 must also be enabled on each of the routers listed in the router lists. You must also enable the specified WCCP service on each of the routers listed in a particular router list that has been associated with a specific WCCP service.
- After enabling WCCP Version 2 on the standalone Content Engine and defining a list of WCCP Version 2-enabled routers that will support this service, you must enable the specific WCCP services on the Content Engine and the WCCP-enabled routers that will be supporting that particular WCCP service.

The following example shows how to enable a user-defined WCCP service (service 91) on a WCCP Version 2-enabled router:

```
Router# configure terminal
Router(config)# ip wccp 91
Router(config)# interface ethernet 0
Router(config-if)# ip wccp 91 redirect out
```

- Use the `show ip wccp` EXEC command on the WCCP Version 2-enabled router to view values associated with WCCP variables.
- The Time To Live (TTL) value of routers servicing a cluster must be 15 seconds or less.
• Service groups consist of up to 32 Content Engines and 32 WCCP-enabled routers.
• All Content Engines in a cluster must include all WCCP-enabled routers that are servicing the cluster in their configuration. If a Content Engine within a cluster does not include one or more of the routers in its configuration, the service group detects the inconsistency and the Content Engine is not allowed to operate within the service group.

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.

Many WCCP Version 2 features also require a configuration of certain options in the `wccp` global configuration command. Refer to the Cisco ACNS Software Command Reference, Release 5.5 publication for more details on the `wccp` global configuration command. 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.

### Configuring Standalone Content Engines to Support User-Defined WCCP Services

A user-defined WCCP service is a WCCP Version 2 service in which port numbers can be configured to redirect traffic to a standalone Content Engine. The Content Engine is functioning as a transparent forward proxy server.

WCCP Version 2 allows you to configure up to eight user-defined WCCP services (services 90 to 97) that support multiple ports (up to eight ports per WCCP service). In order to configure these services, you must create one port list for each service that will be used (for example, create port list number 1 for service 90). The port list contains the port numbers that the WCCP Version 2-enabled router will support WCCP redirection on for that particular user-defined WCCP service. When configuring these user-defined WCCP services, you must specify whether the traffic is to be redirected to the caching application, the HTTPS caching application, or the streaming application on the Content Engine.

To configure the Content Engine to cache web traffic using multiple ports, configure the Content Engine and WCCP Version 2-enabled routers to run a user-defined WCCP service. Use these user-defined web services to support WCCP redirection of HTTP, MMS, HTTPS, and RTSP requests on multiple ports (up to eight ports per service) for the standard WCCP services (for example, the predefined `https-cache`, `rtsp`, `mmst`, and `reverse-proxy` services) that ordinarily only support a single port.

You can use the Content Engine GUI or CLI commands to configure a user-defined WCCP service on a Content Engine. From the Content Engine GUI, select **WCCP > Services**. Use the displayed Services window to configure a Content Engine to support a user-defined WCCP service (services 90 to 97). For more information about how to use the Services window, click the **HELP** button in the window.
From the Content Engine CLI, use the `wccp service-number` global configuration command to configure the Content Engine to support a user-defined WCCP service. As the following example shows, you can define a mask and router list for each user-defined WCCP service (for example, service 95) that you create.

```
ContentEngine(config)# wccp service-number 95 ?
 mask     Specify mask used for CE assignment
 router-list-num Router list number
```

Use the `router-list-num` option to specify the list of WCCP Version 2-enabled routers that the Content Engine should accept redirected requests from for the specified WCCP service.

```
ContentEngine(config)# wccp service-number 95 router-list-num 1
```

**Note** The `router-list-num` option is not required when you are configuring Layer 2 redirection using the `http l4-switch` command or using policy-based routing. (See the “Configuring Layer 4 Switching as a Redirection Method” section on page 6-50.)

Use the `port lists` option to specify which ports the Content Engine is to listen on for redirected traffic from the specified WCCP service. The following example shows how to configure the Content Engine to listen on the ports listed in port list number 5 for redirected traffic for service 95 from the WCCP Version 2-enabled routers on router list 1.

```
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 5
```

Use the `applications` option to specify whether the redirected traffic is to be directed to the caching application, the HTTPS caching application, or the streaming application on the Content Engine.

```
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 1 application ?
 cache     Direct traffic to the caching application
 https-cache Direct traffic to the https caching application
 streaming  Direct traffic to the streaming media application
```

If you use the Content Engine GUI to enable and configure WCCP on the standalone Content Engine, then you must specify the designated router list for each service in each of the following Content Engine GUI windows: the Web Cache window (WCCP > Web Cache), the Reverse Proxy window (WCCP > Reverse Proxy), the Custom Web Cache window (WCCP > Custom Web Cache), and the WCCP Services window (WCCP > Services).

For an example of how to configure user-defined WCCP services (services 90 to 97) for standalone Content Engines, see the “Example 4—Configuring Multiple WCCP Version 2 Services on Standalone Content Engines” section on page 6-46.

---

### Displaying WCCP Configuration Information for Standalone Content Engines

There are several Content Engine CLI commands that you can use to display WCCP-related configuration information (for example, a list of currently configured WCCP services) on standalone Content Engines.

To display a list of the WCCP services that are currently configured on a standalone Content Engine, enter the `show wccp services` EXEC command:

```
ContentEngine# show wccp services
Services configured on this Content Engine
 Web Cache
     Reverse Proxy
```
Disabling and Reenabling WCCP Flow Redirection on Standalone Content Engines

By default, WCCP flow redirection is enabled on a standalone Content Engine.

To reenable WCCP flow redirection on a standalone Content Engine, use the `wccp flow-redirect enable` command.

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

To disable WCCP flow redirection, use the `no` form of this command.

```
ContentEngine(config)# no wccp flow-redirect enable
```

For more information, see the “Configuring WCCP Flow Protection” section on page 15-9.

Shutting Down WCCP on Standalone Content Engines

To prevent broken TCP connections, the Content Engine performs a proper shutdown of WCCP after you enter the `reload` or `no wccp version` command. The Content Engine does not reboot until either all connections have been serviced or the maximum wait time (specified with the `wccp shutdown max-wait` command [by default, 120 seconds]) has elapsed for WCCP Version 2.

During a proper shutdown of WCCP, the Content Engine continues to service the flows it is handling but starts to bypass new flows. When the number of flows goes down to zero, the Content Engine takes itself out of the cluster by having its buckets reassigned to other Content Engines by the lead Content Engine. TCP connections can still be broken if the Content Engine crashes or is rebooted without WCCP being properly shut down.
You cannot shut down an individual WCCP service on a particular port (for example, you cannot shut down the reverse proxy service on port 80) on a Content Engine; you must shut down WCCP on the Content Engine. After WCCP is shut down on the Content Engine, the Content Engine still preserves its WCCP configuration settings and still services proxy-style requests (for example, HTTP requests that the Content Engine receives directly from a client browser).

You can use the Content Engine CLI or GUI to shut down WCCP on a standalone Content Engine. From the Content Engine GUI, check the Clean WCCP shutdown check box in the main window of the Content Engine GUI (see Figure 4-17), and then click the REBOOT button in the same window.

To use the Content Engine CLI to perform a proper shutdown of WCCP on a standalone Content Engine, follow these steps.

---

**Step 1** Specify the time to wait for a proper shutdown.

```
Content Engine(config)# wccp shutdown max-wait seconds
```

*seconds* is the maximum period in seconds (0–86400) that the Content Engine waits before it performs a proper shutdown of WCCP after you have entered a `no wccp version` command. The default is 120 seconds. This command is only supported for WCCP Version 2. The following example shows how to configure the Content Engine to wait 1000 seconds:

```
ContentEngine(config)# wccp shutdown max-wait 1000
```

**Step 2** Shut down WCCP Version 2 on the Content Engine.

```
ContentEngine(config)# no wccp version 2
```

The Content Engine waits 1000 seconds before it shuts down WCCP Version 2. A countdown message appears, indicating how many seconds remain before WCCP will be shut down on the Content Engine.

```
Waiting (999 seconds) for WCCP shutdown. Press ^C to skip shutdown
```

The shutdown can be aborted while in progress by simultaneously pressing ^C after the countdown message appears.

---

**Configuring a Router for WCCP Transparent Redirection**

This section describes how to perform the following tasks on a router:

- Setting a Password for a WCCP Version 2-Enabled Router, page 6-20
- Performing a General WCCP Version 2 Configuration on a Router, page 6-20
- Enabling WCCP on a Router, page 6-20
- Enabling a WCCP Version 2 Router to Support WCCP Service Groups, page 6-20
- Enabling Packet Redirection on Outbound or Inbound Interfaces Using WCCP, page 6-24
- Bypassing the Content Engine with Router Access Lists, page 6-26
Use the **ip wccp** command on the WCCP Version 2-enabled router to specify the WCCP service that you want the router to run. The WCCP service is specified by its service number or name. For a complete list of supported WCCP services, service numbers, and names, see Table B-3. For information about how to configure WCCP services on a router, see the “Configuring WCCP Services on a Router” section on page 6-27.
Setting a Password for a WCCP Version 2-Enabled Router

You must set a password for a WCCP Version 2-enabled router that the standalone Content Engine will access, as follows:

```bash
Router(config)# ip wccp web-cache password [0-7] password
```

where:

- **password** is the string that directs the WCCP Version 2-enabled router to apply MD5 authentication to messages received from the specified service group. Messages that are not accepted by the authentication are discarded.
- **0-7** is the 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** is the optional password name that is combined with the HMAC MD5 value to create security for the connection between the router and the Content Engine.

Performing a General WCCP Version 2 Configuration on a Router

The following example shows a general WCCP Version 2 configuration session on a router:

```bash
Note
You must enter the `ip wccp version 2` command on all WCCP Version 2 router configurations to enable redirection using WCCP Version 2.

Router# configure terminal
Router(config)# ip wccp version 2
Router(config)# interface ethernet0
Router(config-if)# ip wccp web-cache redirect out
```

Enabling WCCP on a Router

To enable WCCP on a router, enter the `ip wccp version` global configuration command. For example, the following command enables WCCP Version 2 on the router:

```bash
Router(config)# ip wccp version 2
```

**Note**
The `ip wccp` global configuration command must be used to enable WCCP on each router that is included on the router list.

Enabling a WCCP Version 2 Router to Support WCCP Service Groups

WCCP Version 2 enables a set of Content Engines in a Content Engine cluster to connect to multiple routers. The Content Engines in a cluster and the WCCP Version 2-enabled routers connected to the Content Engine cluster that are running the same service are known as a *service group*. Standalone Content Engines (Content Engines that are not registered with a Content Distribution Manager) can be part of a Content Engine cluster.
Through communication with the Content Engines, the WCCP Version 2-enabled routers are aware of the available Content Engines. Routers and Content Engines become aware of one another and form a service group using WCCP Version 2. See Figure 6-3.

**Figure 6-3  Service Groups with WCCP Version 2**

Once the service group has been established, one of the Content Engines is designated to determine the lead assignments among the Content Engines in the Content Engine cluster. The type of supported WCCP services varies depending on whether WCCP Version 1 or Version 2 is used. All WCCP services that are listed in Table B-3 except for the standard web-cache service (service 0) require WCCP Version 2.

WCCP uses the concept of a service group to define caching-related services for a WCCP Version 2-enabled router and Content Engines in a cluster. WCCP also redirects user requests from clients requesting these caching-related services to these clusters in real time. In transparent caching through WCCP, you can configure a WCCP Version 2-enabled router to redirect requests to the Content Engine that is functioning as a transparent caching engine.

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

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

In Figure 6-4, the two Content Engines on the left handle only HTTP traffic through port 80 and are defined as members of service group 90. The two Content Engines on the right handle only Microsoft Media Server (MMS) requests through port 99 are defined as members of service group 91.
Chapter 6 Configuring Transparent Redirection for Standalone Content Engines

Figure 6-4 WCCP Version 2 Service Groups

The custom web-cache and reverse-proxy services (service 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.

With eight user-defined services using a maximum number of eight ports each, the maximum number of ports that can be specified for transparent redirection is 64.

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 Content Engines in a cluster).

To direct a WCCP Version 2-enabled router to enable or disable support for a WCCP service group, use the `ip wccp` global configuration command. To remove the ability of a router to control support for a WCCP 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]
```

Table 6-4 describes the `ip wccp` command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-cache</td>
<td>Enables the web-cache service.</td>
</tr>
<tr>
<td>service-number</td>
<td>Identification number of the WCCP service being controlled by a WCCP Version 2-enabled router. The service number can be from 0 to 99. For a list of WCCP service numbers, see Table B-3.</td>
</tr>
<tr>
<td>group-address</td>
<td>(Optional) Directs the WCCP Version 2-enabled router to use a specified multicast IP address for communication with the WCCP service group.</td>
</tr>
<tr>
<td>groupaddress</td>
<td>(Optional) Multicast address used by the WCCP Version 2-enabled router to determine which Content Engine should receive redirected messages.</td>
</tr>
<tr>
<td>redirect-list</td>
<td>(Optional) Directs the WCCP Version 2-enabled router to use an access list to control traffic redirected to this WCCP service group.</td>
</tr>
</tbody>
</table>
The following example shows how to configure a WCCP Version 2-enabled router to run the WCCP reverse proxy service (service 99), using (listening to) the multicast address 172.31.0.0:

```bash
Router(config)# ip wccp 99 group-address 172.31.0.0
```

**Note**

Use the `ip wccp group-listen` command to configure an interface on a WCCP Version 2-enabled router to enable or disable the reception of IP multicast packets for the WCCP feature.

WCCP Version 2 provides authentication that enables you to control which WCCP Version 2-enabled routers and Content Engines become part of the WCCP service group. You use passwords and the HMAC MD5 algorithm by the `ip wccp password [0-7] password` command to control service group membership.

### Enabling WCCP Redirection on the Router

Before you use WCCP Version 2, you must 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.

The enable a router interface to use WCCP Version 2 to redirect web traffic to a standalone Content Engine that is running the ACNS 5.x software, follow these steps:

**Step 1** Enable the router to use WCCP.

```bash
Router# configure terminal
Router(config)# ip wccp version 2
```

**Step 2** (Optional) Specify a redirect access list.

---

**Table 6-4 Parameters of the `ip wccp` Command (continued)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>access-list</code></td>
<td>(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.</td>
</tr>
<tr>
<td><code>group-list</code></td>
<td>(Optional) Directs the WCCP Version 2-enabled router to use an access list to determine which Content Engines are allowed to participate in the WCCP service group.</td>
</tr>
<tr>
<td><code>access-list</code></td>
<td>(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 WCCP service group.</td>
</tr>
<tr>
<td><code>password</code></td>
<td>(Optional) String that directs the WCCP Version 2-enabled router to apply MD5 authentication to messages received from the specified service group. Messages that are not accepted by the authentication are discarded.</td>
</tr>
<tr>
<td><code>0-7</code></td>
<td>(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.</td>
</tr>
<tr>
<td><code>password</code></td>
<td>(Optional) Password name that is combined with the HMAC MD5 value to create security for the connection between the WCCP Version 2-enabled router and the Content Engine.</td>
</tr>
</tbody>
</table>
Chapter 6  Configuring Transparent Redirection for Standalone Content Engines

Configuring a Router for WCCP Transparent Redirection

Only packets that match this access list are redirected to the Content Engine. If you do not specify a redirect access list, all packets are redirected to the Content Engine.

Router(config)# ip wccp redirect-list [number | name]

Step 3  Enter interface configuration mode by specifying an interface name and number.

The following example shows how to specify the Ethernet 0 interface:

Router(config)# interface ethernet 0

Step 4  Configure the router interface that is connected to the Internet to redirect web traffic to the standalone Content Engine.

Router(config-if)# ip wccp web-cache redirect [in | out]

Note  All WCCP-enabled routers support the out option but only certain routers support the in option. Consequently, we recommend that you specify the out option whenever possible. For more information, see the next section, “Enabling Packet Redirection on Outbound or Inbound Interfaces Using WCCP.”

Step 5  (Optional) Configure the router to use the fast switching path on the interface if the client and a Content Engine are located on the same network.

Router(config-if)# ip route-cache same-interface

Step 6  Exit configuration mode.

Router(config-if)# end

Step 7  Save the running configuration to the startup configuration, which is stored in nonvolatile memory.

Router # copy running-config startup-config

Enabling Packet Redirection on Outbound or Inbound Interfaces Using WCCP

Redirection can be specified for either outbound 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 of inbound traffic on interfaces allows you to avoid the overhead associated with CEF forwarding for outbound traffic. If you enable redirection on an outbound interface, this can cause all packets that arrive at all interfaces to take the slower switching path. If you enable redirection on an inbound interface, only those packets arriving at that interface will take the configured feature path; packets arriving at other interfaces will use the faster default path. However, not all routers support redirection on their inbound interfaces.

Configuring WCCP for inbound traffic also allows packets to be classified before the routing table lookup, which provides faster redirection of packets.
To enable packet redirection on an outbound or inbound interface using WCCP, use the `ip wccp redirect` interface configuration command as described in Table 6-5. To disable WCCP redirection, use the `no` form of this command.

```
ip wccp {web-cache | service-number} redirect {out | in}
```

**Table 6-5 Parameters of the ip wccp redirect Command**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-cache</td>
<td>Enables the web-cache service.</td>
</tr>
<tr>
<td>service-number</td>
<td>Identification number of the WCCP service group being controlled by a WCCP Version 2-enabled router. The number can be from 0 to 99. See Table B-3 for a list of supported service numbers.</td>
</tr>
<tr>
<td>redirect</td>
<td>Enables packet redirection checking on an outbound or inbound interface.</td>
</tr>
<tr>
<td>out</td>
<td>Specifies packet redirection on an outbound interface.</td>
</tr>
<tr>
<td>in</td>
<td>Specifies packet redirection on an inbound interface.</td>
</tr>
</tbody>
</table>

The `ip wccp redirect` interface 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.

To disable caching for certain clients, follow these steps:

1. **Step 1** Set the access list used to enable redirection.
   ```
   Router# configure terminal
   Router(config)# ip wccp web-cache redirect-list access-list number
   ```

2. **Step 2** Create an access list that enables or disables traffic redirection to the Content Engine.
   ```
   Router(config)# access-list access-list number deny host host-address
   ```

3. **Step 3** Set the access list to enable access to any host.
   ```
   Router(config)# access-list access-list number permit ip any
   ```

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 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 Ethernet interface 0/1 is redirected to a Content 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
```
Bypassing the Content Engine with Router Access Lists

By default, all HTTP packets are redirected to the Content Engine. A WCCP Version 2-enabled router can be configured with access lists to permit or deny redirection of traffic to a standalone Content Engine. In the following example, traffic conforming to the following criteria are 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
```

Traffic not explicitly permitted is implicitly denied redirection. The `access-list 120 permit ip 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 in 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.

Use the `ip wccp redirect-list` global configuration command to limit the redirection of packets to those matching an access list. Use this command to specify which packets should be redirected to the Content Engine.

When WCCP is enabled but the `ip wccp redirect-list` command is not used, all web-related packets are redirected to the Content Engine. When you specify the `ip wccp redirect-list` command, only packets that match the access list are redirected.

The `ip wccp` global configuration 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 instruct an interface on the WCCP-enabled router to check for appropriate outgoing packets and redirect them to a Content Engine, use the `ip web-cache redirect` interface configuration command. When the `ip wccp` command is enabled but the `ip web-cache redirect` command is disabled, the WCCP-enabled router is aware of the Content Engine but does not use it.

Some websites use the source IP address of packets for authentication. The Content Engine uses its own IP address when sending requests to websites. Thus, the requests from the Content Engine may not be authenticated. Use the `ip wccp redirect-list` command to bypass the Content Engine in these cases.

```
ip wccp redirect-list {number | name}
```

where:

- `number` is the standard or extended IP access list number from 1 to 199.
- `name` is the standard or extended IP access list name. This argument is only available in Cisco IOS Release 11.2 P.
Chapter 6 Configuring Transparent Redirection for Standalone Content Engines

Configuring WCCP Services on a Router

This section describes how to configure the following WCCP services on a router that is running WCCP Version 2:

- Configuring the Standard Web-Cache Service (Service 0) on a Router, page 6-27
- Configuring the DNS Caching Service (Service 53) on a Router, page 6-28
- Configuring the FTP-Native Caching Service (Service 60) on a Router, page 6-29
- Configuring the HTTPS-Cache Service (Service 70) on a Router, page 6-29
- Configuring the RTSP Service (Service 80) on a Router, page 6-30
- Configuring the WMT-RTSPU Service (Service 83) on a Router, page 6-31
- Configuring User-Defined WCCP Services (Services 90–97) on a Router, page 6-31
- Configuring the Custom-Web-Cache Service (Service 98) on a Router, page 6-32
- Configuring the Reverse-Proxy Service (Service 99) on a Router, page 6-33

Remember that after you configure the WCCP service on the router, you must also configure the Content Engine to accept the redirected requests. See “Configuring Standalone Content Engines for WCCP Transparent Redirection” section on page 6-9.

Configuring the Standard Web-Cache Service (Service 0) on a Router

The standard web-cache service (service 0) is a predefined web-caching service that permits a single WCCP Version 1-enabled router or one or more WCCP Version 2-enabled routers to redirect HTTP traffic to standalone Content Engines on port 80 only. In order for a standalone Content Engine to accept redirected HTTP requests on port 80, you must also configure this service on the Content Engine (that is acting as a transparent HTTP forward proxy cache).

The following is an example of how to enable WCCP Version 2 on a router, and then configure the standard web-cache service (service 0) on the router:

Step 1 Enable WCCP Version 2 on the router.

```
Router# configure terminal
Router(config)# ip wccp version 2
```

Step 2 Enable the standard web-cache service (service 0) on the router.

```
Router(config)# ip wccp web-cache
```

Step 3 Specify the interface on which the standard web-cache service will run. Typically, this interface carries the traffic that is going out to the Internet.

```
Router(config)# interface type number
```

In the following example, Ethernet interface 0/1 on the router is configured to run the standard web-cache service:

```
Router(config)# interface ethernet 0/1
```
Step 4  Configure the router to check the HTTP traffic that arrives on the interface that the standard web-cache service is configured on (for example, Ethernet interface 0/1). The router checks this traffic to determine whether it should redirect these packets to the standalone Content Engine. This Content Engine is functioning as a transparent forward proxy server that will accept redirected HTTP requests on port 80 from this WCCP Version 2-enabled router.

Router(config-if)# ip wccp web-cache redirect out

Remember that you must also configure the standalone Content Engine to accept redirected HTTP requests on port 80 by configuring the standard web-cache service on the Content Engine (transparent HTTP forward proxy caching). For more information on this topic, see the “Configuring the Standard Web-Cache Service (Service 0) for Standalone Content Engines” section on page 7-18.

Configuring the DNS Caching Service (Service 53) on a Router

The DNS caching service (service 53) is a predefined WCCP Version 2 caching service. This service permits WCCP Version 2-enabled routers to redirect client requests transparently to a Content Engine so that the Content Engine can resolve the DNS name. After the Content Engine resolves the DNS name, the Content Engine stores it locally so that it can use these resolved names for future DNS requests.

To configure the DNS caching service (service 53) on a router, follow these steps:

Step 1  Enable WCCP Version 2 on the router.

Router# configure terminal
Router(config)# ip wccp version 2

Step 2  Enable the DNS caching service (service 53) on the router.

Router(config)# ip wccp 53

Step 3  Specify the router interface on which the DNS caching service will run.

Router(config)# interface type number

Step 4  Configure the router to use the outbound interface for the DNS caching service.

Router(config-if)# ip wccp 53 redirect out

Remember that you must also configure the DNS caching service (service 53) on the standalone Content Engine before the Content Engine can accept redirected DNS requests from WCCP Version 2 routers. For information on this topic, see the “Configuring DNS Caching for Standalone Content Engines” section on page 7-62.
Configuring the FTP-Native Caching Service (Service 60) on a Router

The ftp-native caching service (service 60) is a predefined WCCP Version 2 caching service. This service permits WCCP Version 2-enabled routers to redirect FTP native requests transparently to a single port on the Content Engine. The Content Engine retrieves the requested FTP content, stores a copy locally (native FTP caching), and serves the requested content to the FTP client.

Note

In the ACNS 5.3.1 software release, the name of this service was changed from “ftp” to “ftp-native” to clearly differentiate between FTP native requests and FTP-over-HTTP requests. Service 60 (the ftp-native caching service) only applies to transparent redirection of FTP native requests and does not apply to FTP-over-HTTP requests.

To configure the ftp-native caching service (service 60) on a router, follow these steps:

Step 1  Enable WCCP Version 2 on the router.
Router# configure terminal
Router(config)# ip wccp version 2

Step 2  Enable the ftp-native caching service (service 60) on the router.
Router(config)# ip wccp 60

Step 3  Specify the interface on which the ftp-native caching service will run.
Router(config)# interface type number

Step 4  Configure the router to use the outbound interface for the ftp-native caching service.
Router(config-if)# ip wccp 60 redirect out

Remember that you must configure the ftp-native caching service on the standalone Content Engine before the Content Engine can accept redirected FTP-native requests from WCCP Version 2 routers. For more information on this topic, see the “Configuring Transparent FTP Native Caching” section on page 7-54.

Configuring the HTTPS-Cache Service (Service 70) on a Router

The https-cache service (service 70) is a predefined WCCP Version 2 web-caching service. This service permits WCCP Version 2-enabled routers to redirect HTTPS traffic transparently to a standalone Content Engine on port 443.

To configure the https-cache service (service 70) on a router, follow these steps:

Step 1  Enable WCCP Version 2 on the router.
Router# configure terminal
Router(config)# ip wccp version 2

Step 2  Enable the https-cache service (service 70) on the router.
Router(config)# ip wccp 70

Step 3  Specify the interface on which the https-cache service will run.
Step 4 Configure the router to use the outbound interface for the https-cache service.

Router(config-if)# ip wccp 70 redirect out

Remember that you must also configure the standalone Content Engine for HTTPS transparent caching before it can accept redirected HTTPS requests from WCCP Version 2-enabled routers. For more information on this topic, see the “Configuring HTTPS Transparent Caching for Standalone Content Engines” section on page 7-27.

**Configuring the RTSP Service (Service 80) on a Router**

The rtsp service (service 80) is a predefined WCCP Version 2 media-caching service. This media-caching service that permits WCCP Version 2-enabled routers to redirect RTSP client requests transparently to a single port on a Content Engine (RealMedia transparent caching).

The Content Engine listens for redirected RTSP requests on the standard RTSP port (default port 554). To intercept RTSP traffic on ports other than the default port (port 554), configure a user-defined WCCP service (services 90 to 97). To configure transparent interception of RTSP requests from RealMedia clients, you only need to configure the rtsp service (service 80) on the WCCP Version 2-enabled router.

In contrast, you must configure the rtsp service (service 80) as well as the wmt-rtspu service (service 83) on the WCCP Version 2-enabled router to configure transparent interception of WMT RTSP requests. For information about configuring service 83 on a router, see the “Configuring the WMT-RTSPU Service (Service 83) on a Router” section on page 6-31.

To configure the rtsp service (service 80) on a router, follow these steps:

**Step 1** Enable WCCP Version 2 on the router.

Router# configure terminal
Router(config)# ip wccp version 2

**Step 2** Enable the rtsp service (service 80) on the router.

Router(config)# ip wccp 80

**Step 3** Specify the interface on which the rtsp service will run.

Router(config)# interface type number

**Step 4** Configure the router to use the outbound interface for the rtsp service.

Router(config-if)# ip wccp 80 redirect out
Chapter 6      Configuring Transparent Redirection for Standalone Content Engines

Configuring WCCP Services on a Router

Configuring the WMT-RTSPU Service (Service 83) on a Router

The wmt-rtspu service (service 83) is a predefined WCCP Version 2 media-caching service. This service permits WCCP Version 2-enabled routers to redirect RTSP client requests from Windows Media 9 players transparently to a single port on a Content Engine (that is, acting as a transparent proxy server, which is configured for WMT RTSP transparent caching). The wmt-rtspu service was added in the ACNS 5.3.1 software release, and is also called the rtspu service.

The Content Engine listens for redirected RTSP requests on the standard RTSPU port (default port 5005). To intercept WMT RTSP traffic on ports other than the default port (port 5005), configure a user-defined WCCP Version 2 service (services 90 to 97).

To configure the wmt-rtspu service (service 83) on a router, follow these steps:

Step 1  Enable WCCP Version 2 on the router.

```
Router# configure terminal
Router(config)# ip wccp version 2
```

Step 2  Enable the wmt-rtspu service (service 83) on the router.

```
Router(config)# ip wccp 83
```

Step 3  Specify the interface on which the wmt-rtspu service will run.

```
Router(config)# interface type number
```

Step 4  Configure the router to use the outbound interface for the wmt-rtspu service.

```
Router(config-if)# ip wccp 82 redirect out
```

You must also configure the rtsp service (service 80) on the WCCP router to support transparent redirection of WMT RTSP requests to a Content Engine.

Remember that you must also configure WMT RTPS transparent caching on the standalone Content Engine before it can accept redirected WMT RTSP requests from WCCP Version 2-enabled routers. For more information on this topic, see the “Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines” section on page 9-14.

Configuring User-Defined WCCP Services (Services 90-97) on a Router

To configure a router to use WCCP Version 2 to support a user-defined WCCP service (services 90 to 97), follow these steps:

Step 1  Enable WCCP Version 2 on the router.

```
Router# configure terminal
```
Chapter 6      Configuring Transparent Redirection for Standalone Content Engines

Configuring WCCP Services on a Router

```
Router(config)# ip wccp version 2
```

**Step 2**
Enable the WCCP feature for the user-defined service (for example, service 90).

```
Router(config)# ip wccp 90
```

**Step 3**
Specify the interface on which service 90 will run on the router.

```
Router(config)# interface type number
```

**Step 4**
Configure the Content Engine to use the outbound interface for service 90.

```
Router(config-if)# ip wccp 90 redirect out
```

Remember that you must also configure the user-defined WCCP service (for example, service 90) on the standalone Content Engine before the Content Engine can accept redirected proxy packets from WCCP Version 2-enabled routers. For more information on this topic, see the “Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 6-15.

Configuring the Custom-Web-Cache Service (Service 98) on a Router

The custom-web-cache service (service 98) is a predefined WCCP Version 2 web-caching service. This service permits WCCP Version 2-enabled routers to redirect HTTP traffic to a Content Engine on multiple ports other than port 80. The Content Engine is functioning as a transparent forward proxy server. This WCCP service allows you to configure the Content Engine to listen on multiple ports (up to eight ports) for WCCP redirected HTTP requests without having to configure a user-defined WCCP service (services 90 to 97).

To configure a router to use WCCP Version 2 to support the custom-web-cache service (service 98), follow these steps:

**Step 1**
Enable WCCP Version 2 on the router.

```
Router# configure terminal
Router(config)# ip wccp version 2
```

**Step 2**
Enable the custom-web-cache service (service 98) on the router.

```
Router(config)# ip wccp 98
```
Chapter 6  Configuring Transparent Redirection for Standalone Content Engines

Configuring WCCP Services on a Router

**Step 3**  Specify the interface on which the custom-web-cache service will run.

Router(config)# interface type number

In the following example, the Ethernet 0 interface is configured to run the custom-web-cache service:

Router(config)# interface ethernet 0

**Step 4**  Configure the router to use the outbound interface for the custom-web-cache service.

Router(config-if)# ip wccp 98 redirect out

Remember that you must also configure the custom-web-cache service on the standalone Content Engine before the Content Engine can accept redirected web cache proxy packets from WCCP Version 2-enabled routers on multiple ports. For more information on this topic, see the “Configuring the Custom Web-Cache Service (Service 98) for Standalone Content Engines” section on page 7-20.

**Configuring the Reverse-Proxy Service (Service 99) on a Router**

The reverse-proxy service (service 99) is a predefined WCCP Version 2 service. This service permits WCCP Version 2-enabled routers to redirect reverse proxy packets to a standalone Content Engine that is functioning as a transparent reverse proxy server.

To configure a router to use WCCP Version 2 to support the reverse-proxy service (service 99), follow these steps:

**Step 1**  Enable WCCP Version 2 on the router.

Router# configure terminal
Router(config)# ip wccp version 2

**Step 2**  Enable the reverse-proxy service (service 99) on the router.

Router(config)# ip wccp 99

**Step 3**  Specify the interface on which the reverse-proxy service will run.

Router(config)# interface type number

In the following example, the Ethernet 0 interface is configured to run the reverse-proxy service:

Router(config)# interface ethernet 0

**Step 4**  Configure the router to use the outbound interface for the reverse-proxy service. The router will check the reverse proxy packets on Ethernet interface 0 to determine if it should transparently redirect these packets to the Content Engine (that is acting as a transparent reverse proxy server).

Router(config-if)# ip wccp 99 redirect out

Remember that you must also configure the reverse-proxy service (service 99) on the standalone Content Engine before the Content Engine can accept redirected reverse proxy packets from WCCP Version 2 routers. For more information on this topic, see the “Configuring HTTP Reverse Proxy Caching for Standalone Content Engines” section on page 7-23.
Clearing WCCP Statistics on a Router

Use the `clear ip wccp` EXEC command to clear the WCCP statistics maintained on the WCCP Version 2 router, either for a particular service or for all the services.

```
clear ip wccp {web-cache | service-number}
```

where:

- `web-cache` specifies that the router should remove statistics for the web-cache service.
- `service-number` specifies that the router should remove statistics for the specified service. The service group number can be from 0 to 99.

For example, direct the router to clear the statistics for the reverse-proxy service (service 99) by entering the following command on the WCCP Version 2-enabled router:

```
Router# clear ip wccp web-cache 99
```

Configuring WCCP Layer 2 Support

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. This allows the Content Engine to perform a Layer 2 or MAC address rewrite redirection if it is directly connected to a compatible Cisco switch. This redirection processing is accelerated in the switching hardware, which makes this method a more efficient method than Layer 3 redirection using GRE.

The Content Engine must have a Layer 2 connection with the switch. Because there is no requirement for a GRE tunnel between the switch and the Content Engine, the switch can use a cut-through method of forwarding encapsulated packets using the `l2-redirect` option in the CLI.

Two load-balancing schemes exist between WCCP Version 2-enabled routers or switches and Content Engines when the Layer 2 forwarding method is chosen:

- Hash assignment
  - For the Catalyst 6000 and 6500 series switches, this load-balancing method is called WCCP Layer 2 Policy Feature Card (PFC) redirection. 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).

- Mask assignment
  - This type of load-balancing is called the WCCP Layer 2 Policy Feature Card 2 (PFC2) redirection. It uses a combination of the Supervisor Engine 2 and the MSFC2.

You can use the Content Engine GUI or CLI commands to specify the load-balancing schemes for a specific WCCP service on a Content Engine. All WCCP services supported by the Content Engine use such CLI commands as `wccp custom-web-cache`, `wccp media-cache`, `wccp reverse-proxy`, `wccp service-number`, `wccp web-cache`, `wccp wmt`, and `wccp rtsp` to support either the hash or the mask assignment load-balancing method with Layer 2 forwarding. You can specify one load-balancing method (hashing and masking) per WCCP service in a Content Engine cluster. For example, if you define three WCCP services for Content Engine Cluster A, two of the services in Cluster A could be using the hash load-balancing method. The third service in Cluster A could be using the mask load-balancing method.
Note
You can only enable Layer 2 redirection with the mask assignment load-balancing method through the Content Engine CLI (this is not supported through the Content Engine GUI).

For information about how to configure the hash load-balancing method, see the next section, “Configuring Layer 2 Forwarding with the Hash Load-Balancing Method.” For information about how to configure the mask load-balancing method, see the “Configuring Layer 2 Forwarding with the Mask Load-Balancing Method” section on page 6-36.

Configuring Layer 2 Forwarding with the Hash Load-Balancing Method

Both types of packet-forwarding methods (layer GRE and layer 2 redirection) support hashing as a load-balancing method. Hashing allows you to specify how redirected traffic should be load balanced among multiple Content Engines in a Content Engine cluster.

When configuring user-defined WCCP Version 2 services on a Content Engine, you can configure hashing parameters (for example, hash on source IP address) for that particular user-defined WCCP Version 2 service. The default hashing assignment for user-defined WCCP services (Services 90 to 97) is hash on destination IP addresses. You use the `wccp service-number` global configuration command to change the default hashing assignment for any of the user-defined WCCP services.

The following example shows how you can use the `wccp service-number` command to configure a user-defined WCCP service (in this case, Service 90) to hash on source IP addresses instead of destination IP addresses:

```
ContentEngine(config)# wccp service-number 90 router-list-num 1 port-list-num 1
application cache hash-source-ip
```

Table 6-6 lists the default hashing assignments. The default hashing assignment for predefined WCCP services is fixed and cannot be changed.

**Table 6-6  Default Hashing Assignments for WCCP Version 2 Services**

<table>
<thead>
<tr>
<th>WCCP Service Type</th>
<th>Service Number (Identifier)</th>
<th>Default Hashing Assignment for Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined WCCP services</td>
<td>90 to 97</td>
<td>Default hashing assignment is hash on destination IP addresses</td>
</tr>
<tr>
<td>Predefined WCCP services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>web-cache</td>
<td>0</td>
<td>Hash on destination IP addresses</td>
</tr>
<tr>
<td>dns caching</td>
<td>53</td>
<td>Hash on source port</td>
</tr>
<tr>
<td>ftp-native</td>
<td>60</td>
<td>Hash on destination IP addresses</td>
</tr>
<tr>
<td>https-cache</td>
<td>70</td>
<td>Hash on source IP addresses</td>
</tr>
<tr>
<td>rtsp</td>
<td>80</td>
<td>Hash on destination IP addresses</td>
</tr>
<tr>
<td>mmst</td>
<td>81</td>
<td>Hash on destination IP addresses</td>
</tr>
<tr>
<td>mmsu</td>
<td>82</td>
<td>Hash on destination IP addresses</td>
</tr>
<tr>
<td>wmt-rtspu</td>
<td>83</td>
<td>Hash on destination IP addresses</td>
</tr>
</tbody>
</table>
Configuring WCCP Layer 2 Support

Chapter 6 Configuring Transparent Redirection for Standalone Content Engines

Table 6-6 Default Hashing Assignments for WCCP Version 2 Services (continued)

<table>
<thead>
<tr>
<th>WCCP Service Type</th>
<th>Service Number (Identifier)</th>
<th>Default Hashing Assignment for Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom-web-cache</td>
<td>98</td>
<td>Hash on destination IP addresses</td>
</tr>
<tr>
<td>reverse-proxy</td>
<td>99</td>
<td>Hash on source IP addresses and source port</td>
</tr>
</tbody>
</table>

The following example shows how to configure a Content Engine to receive Layer 2 redirected traffic from a Catalyst 6500 series switch with a Multilayer Switch Feature Card (MSFC) and Supervisory Engine 1A (SUP 1A) using a hash assignment method for load balancing. To configure the Content Engine, follow these steps:

Step 1 Enable WCCP Version 2 on the Content Engine.

```
ContentEngine# configure terminal
ContentEngine(config)# wccp version 2
```

Step 2 Create a router list on the Content Engine. In the following example, router list 1 is created and contains only a single WCCP Version 2-enabled router (the router with an IP address of 172.16.55.1).

```
ContentEngine(config)# wccp router-list 1 172.16.55.1
```

Step 3 Configure the standard web-cache service (service 0) on the Content Engine.

Configure this WCCP service to use the router list you just created in Step 2. Enter the `l2-redirect` option to specify Layer 2 redirection as the packet-forwarding method (as opposed to GRE) for this service. Because the mask assignment method is not specified, the default hash assignment method is used to load balance redirected requests.

```
ContentEngine(config)# wccp web-cache router-list-num 1 l2-redirect
```

Step 4 Use the `show wccp services detail` EXEC command to display the configuration so that you can verify it.

```
ContentEngine# show wccp services detail
```

Step 5 Write the running configuration to nonvolatile memory.

```
ContentEngine# copy running-config startup-config
```

Configuring Layer 2 Forwarding with the Mask Load-Balancing Method

Both types of packet-forwarding methods (GRE and Layer 2 redirection) support masking as a load-balancing method. Use the `wccp service-name mask` global configuration command to specify the different masks (for example, the destination IP mask) on the Content Engine.

Use the `wccp service-name mask` global configuration command to change the default masks for a particular WCCP Version 2 service. For example, use the `wccp https-cache mask` global configuration command to configure masks for transparently redirected HTTPS requests (the https-cache service).

```
ContentEngine(config)# wccp https-cache mask?
dst-ip-mask Specify sub-mask used in packet destination-IP address
dst-port-mask Specify sub-mask used in packet destination-port number
src-ip-mask Specify sub-mask used in packet source-IP address
src-port-mask Specify sub-mask used in packet source-port number
```
wccp https-cache \{ mask \{ [dst-ip-mask hex_num] [dst-port-mask port_hex_num] [src-ip-mask hex_num] [src-port-mask port_hex_num] \}\}

Table 6-7 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>Sets the mask used for Content Engine assignment. Configure at least one mask. You can configure up to four masks.</td>
</tr>
<tr>
<td>dst-ip-mask</td>
<td>(Optional) Sets the mask used to match the destination IP address of the redirected packet.</td>
</tr>
<tr>
<td>hex_num</td>
<td>IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000 to FC000000.</td>
</tr>
<tr>
<td>dst-port-mask</td>
<td>(Optional) Sets the mask used to match the destination port number of the redirected packet.</td>
</tr>
<tr>
<td>port_hex_num</td>
<td>Source port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0 to 65535.</td>
</tr>
<tr>
<td>src-ip-mask</td>
<td>(Optional) Sets the mask used to match the source IP address of the redirected packet.</td>
</tr>
<tr>
<td>hex_num</td>
<td>IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000 to FC000000.</td>
</tr>
<tr>
<td>src-port-mask</td>
<td>(Optional) Sets the mask used to match the source port number of the redirected packet.</td>
</tr>
<tr>
<td>port_hex_num</td>
<td>Source port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0 to 65535.</td>
</tr>
</tbody>
</table>

To view the mask configuration for a specific WCCP Version 2 service, use the `show wccp masks service-name` EXEC command:

```
ContentEngine(config)# show wccp masks ?
custom-web-cache Custom web caching service
dns DNS caching service
ftp-native Native FTP caching service
https-cache HTTPS caching service
reverse-proxy Reverse Proxy web caching service
rtsp Media caching service
service-number Custom-service number
web-cache Standard web caching service
wmt-rtspu WMT RTSPU service
```
The following example shows how to configure a Content Engine to receive Layer 2 redirected traffic from a Catalyst 6500 series switch with a Multilayer Switch Feature Card 2 and Supervisor Engine 2 (MSFC2/SUP 2). To configure the Content Engine, follow these steps:

---

Step 1  Enable WCCP Version 2 on the Content Engine.
```
ContentEngine# configure terminal
ContentEngine(config)# wccp version 2
```

Step 2  Create a router list on the Content Engine. In the following example, router list 1 is created and contains only a single WCCP Version 2-enabled router (the router with an IP address of 172.16.55.1):
```
ContentEngine(config)# wccp router-list 1 172.16.55.1
```

Step 3  Configure the web-cache service on the Content Engine. Configure this WCCP service to use the router list created in Step 2. Enter the `l2-redirect` option to specify Layer 2 redirection as the packet-forwarding method (as opposed to GRE). Enter the `mask-assign` option to specify mask assignment as the load-balancing method (as opposed to the default hash assignment method) for this WCCP service.
```
ContentEngine(config)# wccp web-cache router-list-num 1 l2-redirect mask-assign
```

Step 4  Display the configuration so that you can verify it.
```
ContentEngine# show wccp services detail
```

Step 5  Write the running configuration to nonvolatile memory.
```
ContentEngine# copy running-config startup-config
```

---

Examples of Configuring WCCP Services for Standalone Content Engines

The section provides the following examples of how to configure WCCP services for standalone Content Engines using WCCP Version 2:

- Example 1—Configuring the Web-Cache Service with WCCP Version 1, page 6-39
- Example 2—Configuring the Web-Cache Service with WCCP Version 2, page 6-43
- Example 3—Configuring the HTTPS Transparent Caching Service with WCCP Version 2, page 6-45
- Example 4—Configuring Multiple WCCP Version 2 Services on Standalone Content Engines, page 6-46
Note the following important points when configuring the standard web-cache service (service 0) with either WCCP Version 1 or WCCP Version 2:

- The Content Engines must not have their packets encrypted or compressed and should be part of the “inside” Network Address Translation (NAT) firewall if one is present.
- 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.
- Placing the 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.
- You can also use the Content Engine GUI to configure the standard web-cache service on the Content Engine. However, you must always use the CLI to configure the standard web-cache service on the router.
- To use a WCCP-enabled router to support the web cache service, an IP address must be configured on the interface connected to the Internet, and that interface must be connected to the Content Engine. Use the show ip interface EXEC command on the router to see whether the interfaces on the router that are configured for IP are currently usable.

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. Entering the show ip interface EXEC command on an asynchronous interface encapsulated with PPP or SLIP displays a message indicating that IP fast switching is enabled.

**Example 1—Configuring the Web-Cache Service with WCCP Version 1**

You can configure a single WCCP-enabled router and one or more Content Engines to run the standard web-cache service (service 0) using WCCP Version 1. Figure 6-5 shows a sample WCCP Version 1 network configuration that consists of a cluster of three Content Engines that are being serviced by a single WCCP Version 1-enabled router.
With WCCP Version 1, only a single WCCP-enabled router services a Content Engine cluster, becoming the default home router for the cluster. With WCCP Version 1, this single router that is servicing the cluster is the device that performs all the IP packet redirection.

When WCCP Version 1 is used, the following sequence of events occurs between the Content Engines and the single WCCP-enabled router (home router) that services these Content Engines:

1. Each Content Engine records the IP address of the WCCP-enabled router servicing the Content Engine cluster.

2. The Content Engines then transmit their IP addresses to the WCCP-enabled router, indicating their presence to one another in the Content Engine cluster.

3. The WCCP-enabled 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 WCCP-enabled router how IP packet redirection should be performed.

   The lead Content Engine is defined as the one that has the lowest IP address in the cluster, and is seen by the WCCP Version 1-enabled router (home router) that is servicing the cluster.

The following example describes how to use the Content Engine CLI to enable and configure the standard web-cache service (service 0) using WCCP Version 1. With WCCP Version 1, you can only configure one service (the standard web-cache service) and a single WCCP-enabled router (home router). Even if there is a cluster of Content Engines, only a single WCCP Version 1-enabled router services a cluster of Content Engines, becoming the default home router for the cluster.
In this example, IP access lists are used to control which web-related packets are redirected to the standalone Content Engine. This example also shows how you can verify that the web-cache service is operating properly after it has been configured on a single router and the standalone Content Engine.

**Step 1** To use a WCCP-enabled router for the standard web-cache service, an IP address must be configured on the interface connected to the Internet, and the interface must be connected to the standalone Content Engine.

Use the `show ip interface` EXEC command on the router to check whether the interfaces on the router that are configured for IP are usable.

**Step 2** Determine whether WCCP is currently enabled on the router by entering the `show ip wccp` EXEC command.

**Step 3** Enable the router to use WCCP Version 1, and then configure the router to use WCCP Version 1 to redirect web-related packets that do not have a destination of 192.168.196.51 to the standalone Content Engine.

\[
\text{Router}\# \text{configure terminal} \\
\text{Router(config)# ip wccp version 1} \\
\text{Router(config)# access-list 100 deny ip any host 192.168.196.51} \\
\text{Router(config)# access-list 100 permit ip any any} \\
\text{Router(config)# ip wccp web-cache redirect-list 100} \\
\text{Router(config)# interface Ethernet 0} \\
\text{Router(config)# ip wccp web-cache redirect out} \\
\text{Router(config-if)# end} \\
\text{Router# %SYS-5-CONFIG_I: Configured from console by console.}
\]

**Step 4** Enable WCCP Version 1 on the standalone Content Engine.

\[
\text{ContentEngine# configure terminal} \\
\text{ContentEngine(config)# wccp version 1}
\]

**Step 5** Point the standalone Content Engine to the home router by entering the `wccp home-router ip-address` global configuration command. This may also be the IP address of the IP default gateway.

In the following example, the home router has an IP address of 192.168.51.102:

\[
\text{ContentEngine(config)# wccp home-router 192.168.51.102}
\]

**Note** You can also use the Content Engine GUI (choose **WCCP > Enable WCCP**) to configure the WCCP Version 1 web cache service on a standalone Content Engine. For more information about the Enable WCCP window, click the **HELP** button in the window.

**Step 6** Configure the standard web-cache service on the Content Engine, as described in the “Configuring the Standard Web-Cache Service (Service 0) for Standalone Content Engines” section on page 7-18.

**Step 7** Verify that the standard web-cache service is now enabled on this standalone Content Engine by entering the `show wccp` EXEC command.

\[
\text{ContentEngine# show wccp services} \\
\text{Services configured on this Content Engine} \\
\text{Web Cache} \\
\text{ContentEngine#}
\]
You can also display other WCCP information on the standalone Content Engine, by using other options of the `show wccp` EXEC command on the Content Engine. For example, display WCCP generic routing encapsulation packet-related information on the Content Engine, by specifying the `gre` option of the `show wccp` command.

**Step 8** Verify that WCCP is enabled on the router and that the router is aware of the standalone Content Engine that you have configured as a web cache by entering the `show ip wccp web-cache` command on the WCCP-enabled router.

In the following example, the `show ip wccp web-cache` command is entered immediately after the home router has been configured. After a few seconds, the state of the standalone Content Engine that has an IP address of 192.168.25.3 changes from “NOT Usable” to “Usable,” as seen in the second output.

```
Router# show ip wccp web-cache
WCCP Web-Cache information:
    IP Address: 192.168.25.3
    Protocol Version: 1.0
    State: NOT Usable
    Initial Hash Info: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    Assigned Hash Info: 00000000000000000000000000000000
                        00000000000000000000000000000000
    Hash Allotment: 0 (0.00%)  Packets Redirected: 0
    Connect Time: 00:00:06

Router# show ip wccp web-cache
WCCP Web-Cache information:
    IP Address: 192.168.25.3
    Protocol Version: 0.3
    State: Usable
    Initial Hash Info: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    Assigned Hash Info: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    Hash Allotment: 256 (100.00%)  Packets Redirected: 0
    Connect Time: 00:00:31
```

**Step 9** On the WCCP Version 1-enabled router, enter the `debug ip wccp events` EXEC command to view information about significant WCCP events.

The following example is sample output of the `debug ip wccp events` EXEC command when a Content Engine is added to the list of available web caches for this WCCP Version 1-enabled router:

```
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
```

**Step 10** On the WCCP Version 1-enabled router, enter the `debug ip wccp packets` command to view information about every WCCP packet that was received or sent by this router.

The following example is sample output of the `debug ip wccp packets` command. The router is sending keepalive packets to the standalone Content Engine at 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
```
Examples of Configuring WCCP Services for Standalone Content Engines

Examples of Configuring WCCP Services for Standalone Content Engines

Tip
To clear the router’s counter for packets redirected by WCCP, enter the `clear ip wccp` EXEC command on the WCCP-enabled router.

---

Example 2—Configuring the Web-Cache Service with WCCP Version 2

The following example shows how to use the Content Engine CLI to configure the standard web-cache service (service 0) when the clients and the standalone Content Engine are on the same subnet and WCCP Version 2 is being used instead of WCCP Version 1:

Step 1
Ensure that WCCP Version 2 is enabled on each router that will be added to the router list in Step 3.

```
Router(config)# ip wccp version 2
```

Step 2
Enter the `ip wccp web-cache password` global configuration command to configure the standard web-cache service on the WCCP-enabled router, and set a password for this router.

```
Router(config)# ip wccp web-cache [password [0-7] [password]]
```

where:

- **password** directs the WCCP-enabled router to apply MD5 authentication to messages received from the specified service group. Messages that are not accepted by the authentication are discarded.
- 0-7 is an optional value that indicates that the HMAC MD5 algorithm is used to encrypt the password. This value is generated when an encrypted password is created for the Content Engine.
- *password* is the optional password name that is combined with the HMAC MD5 value to create a secure connection between the WCCP-enabled router and the Content Engine.

Step 3
On the standalone Content Engine that you want to configure for the standard web-cache service (HTTP transparent caching using WCCP Version 2), create a router list (for example, router-list 1).

```
ContentEngine(config)# wccp router-list 1 10.10.10.1
```

Step 4
Enable the standard web-cache service on the Content Engine, and specify which WCCP Version 2-enabled routers (router list) will support this particular service for this Content Engine (that is, will redirect HTTP requests to this Content Engine on port 80).

Associate this WCCP service with the router list you just created. Assign the Layer 2 redirection option. If the mask assignment method is not specified, the default load-balancing method is the hash assignment method.

```
ContentEngine(config)# wccp web-cache router-list-num 1 l2-redirect
```
Tip

You can also use the Content Engine GUI (choose WCCP > Web Cache from the Content Engine GUI) to configure the WCCP Version 2 web cache service on a standalone Content Engine. If you use the Content Engine GUI to configure WCCP Version 2 on a Content Engine, then you must specify the designated router list for each WCCP service in each of the following Content Engine GUI windows: the Web Cache window (WCCP > Web Cache), the Reverse Proxy window (WCCP > Reverse Proxy), the Custom Web Cache window (WCCP > Custom Web Cache), and the WCCP Services window (WCCP > Services).

Step 5  Enable WCCP Version 2 on the Content Engine.

    ContentEngine(config)# wccp version 2

Step 6  Exit global configuration mode.

    ContentEngine(config)# exit

Step 7  Write the running configurations to nonvolatile memory on the Content Engine.

    ContentEngine# write memory

Step 8  Now that the router has been configured to run WCCP Version 2, monitor WCCP on the router.

    Router# show ip wccp

or

    Router# show ip wccp {web-cache | 90-99}

Step 9  Query the WCCP-enabled router for information about the Content Engines that the router has detected in a specific service group. The information can be displayed for WCCP services ranging in value from 90 to 99.

    Router# show ip wccp {web-cache | 90-99} detail

Step 10 Determine whether any ip wccp direct commands are configured on an interface.

    Router# show ip interface

Step 11 Display which devices in a particular WCCP service group were 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 90 to 99.

    Router# show ip wccp {web-cache | 90-99} view
Example 3—Configuring the HTTPS Transparent Caching Service with WCCP Version 2

The following example shows how to use the Content Engine CLI to configure a user-defined WCCP Version 2 service and HTTPS transparent caching on a standalone Content Engine. In this example, service number 95 is the user-defined service.

**Step 1** Configure service 95 so that the Content Engine will accept HTTPS requests that are being transparently intercepted and redirected to it by multiple WCCP Version 2-enabled routers on ports other than the default port.

You must first create a router list and a port list that service 95 is to use (for example, create port list 1 for service 95). The port list contains the port numbers that the WCCP Version 2-enabled router will support WCCP redirection for service 95.

```
ContentEngine# configure terminal
ContentEngine(config)# wccp service-number 95 router-list-num 1
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 1
```

**Step 2** When configuring service 95, you must specify whether the traffic is to be redirected to the caching application, the HTTPS caching application, or the streaming application on the Content Engine.

```
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 1 application ?
  cache     Direct traffic to the caching application
  https-cache Direct traffic to the HTTPS caching application
  streaming  Direct traffic to the streaming media application
```

**Step 3** Specify that the Content Engine is to accept traffic that is redirected to its HTTPS application. The WCCP Version 2-enabled routers on router list number 1 will redirect HTTPS traffic to the HTTPS caching application on the Content Engine. The Content Engine will listen for such WCCP redirected requests on the ports that are specified in port list 1.

```
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 1 https-cache
```

In the ACNS 5.2.1 software and later releases, the accept-all mode is supported for the https-cache service. The accept-all mode supports the filtering of HTTPS traffic. This mode works the same way as the traditional WCCP services (for example, the web-cache service that intercepts all web traffic by default).

By default, the Content Engine accepts all HTTPS traffic.

```
ContentEngine(config)# wccp https-cache ?
  accept-all Accept all HTTPS traffic by default
  mask Specify mask used for CE assignement
  router-list-num Router list number
```

**Note** If the `wccp https-cache accept-all` global configuration command is used, the HTTPS cache (the Content Engine that is configured for HTTPS transparent caching) will work in accept all mode (it will intercept all HTTPS traffic); otherwise, the HTTPS cache works in accept only mode as in the ACNS 5.1.x software.
Step 4 | Enable WCCP Version 2 on the Content Engine.

```
ContentEngine(config)# wccp version 2
```

---

**Example 4—Configuring Multiple WCCP Version 2 Services on Standalone Content Engines**

The following example shows how to use the Content Engine CLI to configure 16 WCCP Version 2 services on a standalone Content Engine:

---

**Step 1** Configure a router list that lists the WCCP Version 2-enabled routers that will support the 16 WCCP Version 2 services (eight user-defined services and eight predefined services). In this case, router list 1 has only a single router (the WCCP Version 2-enabled router with an IP address of 10.1.202.1)

```
ContentEngine(config)# wccp router-list 1 10.1.202.1
```

**Step 2** Configure eight port lists (port lists number 1 through 8).
These port lists specify the port numbers on which the Content Engine will listen for incoming traffic from specific WCCP Version 2-enabled routers. These port lists allow you to configure the Content Engine to listen for incoming WCCP requests on more than one port. By default, the Content Engine listens for incoming traffic on port 80. Create one port list for each of the eight user-defined WCCP Version 2 services that you will be creating (services 90 to 97). You can define up to eight ports per port list. In this case, each port list has a single port (for example, port list 1 contains only port 32).

```
ContentEngine(config)# wccp port-list 1 32
ContentEngine(config)# wccp port-list 2 33
ContentEngine(config)# wccp port-list 3 34
ContentEngine(config)# wccp port-list 4 35
ContentEngine(config)# wccp port-list 5 36
ContentEngine(config)# wccp port-list 6 37
ContentEngine(config)# wccp port-list 7 38
ContentEngine(config)# wccp port-list 8 39
```

**Step 3** Enable the standard web-cache service (service 0) on the Content Engine, and associate router list 1 with this first predefined WCCP service.

```
ContentEngine(config)# wccp web-cache router-list-num 1
```

The Content Engine will listen on port 80 for redirected HTTP requests from the routers on router list 1.

*Note* The term HTTP requests is used to refer collectively to HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests.

**Step 4** Enable the reverse-proxy caching service (service 99) on the Content Engine, and associate router list 1 with this second predefined WCCP Version 2 service.

```
ContentEngine(config)# wccp reverse-proxy router-list-num 1
```

The Content Engine will listen on port 80 for redirected reverse proxy requests from the routers on router list 1.
Step 5  Enable the custom-web-cache service (service 98) on the Content Engine, and associate router list 1 and port 31 with this third predefined WCCP Version 2 service.

```
ContentEngine(config) # wccp custom-web-cache router-list-num 1 port 31
```

The Content Engine will listen on port 31 for redirected HTTP requests from the WCCP Version 2-enabled routers on router list 1.

Step 6  Enable the rtsp service (service 80) on the Content Engine, and associate router list 1 with this fourth predefined WCCP Version 2 service.

```
ContentEngine(config) # wccp rtsp router-list-num 1
```

The Content Engine will listen on the standard RTSP port (default port 554) for redirected RTSP requests from the WCCP Version 2-enabled routers on router list 1.

Step 7  Enable the WMT services (services 81 and 82) on the Content Engine, and associate router list 1 with this fifth predefined WCCP Version 2 service.

```
ContentEngine(config) # wccp wmt router-list-num 1
```

After specifying this command, the Content Engine will listen on the default port (port 1755) for redirected WMT requests from the WCCP Version 2-enabled routers on router list 1.

Step 8  Enable the dns caching service (service 53) on the Content Engine, and associate router list 1 with this sixth predefined WCCP Version 2 service.

```
ContentEngine(config) # wccp dns router-list-num 1
```

The Content Engine will listen on port 80 for redirected DNS requests from the WCCP Version 2-enabled routers on router list 1.

Step 9  Enable the ftp-native caching service (service 60) on the Content Engine, and associate router list 1 with this seventh predefined WCCP Version 2 service.

```
ContentEngine(config) # wccp ftp-native router-list-num 1
```

The Content Engine will listen on port 80 for redirected FTP native requests from the WCCP Version 2-enabled routers on router list 1. This is for FTP native caching (as opposed to FTP-over-HTTP caching that is involved when the Content Engine receives FTP-over-HTTP requests directly from a client browser and caches the requested content).

Step 10 Enable the https-caching service (service 70) on the Content Engine, and associate router list 1 with this eighth predefined WCCP Version 2 service.

```
ContentEngine(config) # wccp https-cache router-list-num 1
```

In the ACNS 5.1 software, HTTPS requests could only be SSL-terminated on the Content Engine in WCCP mode. In the ACNS 5.1 software, only HTTPS requests to specific sites (HTTPS origin servers that the Content Engine was specifically configured to support) were SSL-terminated in WCCP mode. In the ACNS 5.1 software, the Content Engine would bypass HTTPS requests that were directed to HTTPS servers that it had not been explicitly configured to support. For more information about SSL termination, see the “About SSL Termination of HTTPS Client Requests” section on page 7-25.

In the ACNS 5.1.x software, only one interception mode (the accept-only mode) was supported for the https-cache service. With the accept-only mode, you had to configure the Content Engine to accept only redirected requests that were directed to specific HTTPS servers, as follows:

```
ContentEngine(config) # wccp https-cache router-list-num 1
```

or
Examples of Configuring WCCP Services for Standalone Content Engines

Chapter 6  Configuring Transparent Redirection for Standalone Content Engines

ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list 1 https-cache

In both of the preceding examples, the Content Engine will only accept the redirected HTTPS traffic if the HTTPS server is configured on the Content Engine (using the **https server** global configuration command).

In the ACNS 5.2.1 software and later releases, the Content Engine SSL terminates HTTPS requests in WCCP mode and in manual proxy mode if the requested HTTPS servers are configured on the Content Engine, and tunnels the rest of the HTTPS traffic. For more information about tunneling of HTTPS requests, see the “About Tunneling of HTTPS Client Requests” section on page 7-25. For specific requested content to be cached, you must import the proper certificates and keys for these HTTPS servers into the Content Engine and configure the Content Engine to cache these servers. For standalone Content Engines, this is performed through the Content Engine CLI, as described in the “Configuring Certificates and Private Keys for HTTPS Caching” section on page 7-32.

**Step 11** Enable the first user-defined WCCP service (service 90) on the Content Engine, and associate router list 1 and port list 1 with this service. Specify that the traffic is to be redirected to the caching application on the Content Engine by entering the **application cache** option.

ContentEngine(config)# wccp service-number 90 router-list-num 1 port-list-num 1 application cache

The Content Engine will listen on the ports listed in port list 1 (port 32) for redirected requests from the routers on router list 1.

**Tip** You must specify the **application cache** option for each user-defined WCCP services (services 90 to 97) that are created in this example because you want the WCCP routers to redirect the traffic to the caching application (as opposed to the streaming application) on the Content Engine.

**Step 12** Enable the second user-defined WCCP service (service 91) on the Content Engine, and associate router list 1 and port list 2 with this service.

ContentEngine(config)# wccp service-number 91 router-list-num 1 port-list-num 2 application cache

The Content Engine will listen on the ports listed in port list 2 (port 33) for redirected requests from the WCCP Version 2-enabled routers in router list 1.

**Step 13** Enable the third user-defined WCCP service (service 92) on the Content Engine, and associate router list 1 and port list 3 with this service.

ContentEngine(config)# wccp service-number 92 router-list-num 1 port-list-num 3 application cache

The Content Engine will listen on the ports listed in port list 3 (port 34) for redirected requests from the routers on router list 1.

**Step 14** Enable the fourth user-defined WCCP service (service 93) on the Content Engine, and associate router list 1 and port list 4 with this service.

ContentEngine(config)# wccp service-number 93 router-list-num 1 port-list-num 4 application cache

The Content Engine will listen on the ports listed in port list 4 (port 35) for redirected requests from the routers on router list 1.
Step 15  Enable the fifth user-defined WCCP service (service 94) on the Content Engine, and associate router list 1 and port list 5 with this service.

```
ContentEngine(config)# wccp service-number 94 router-list-num 1 port-list-num 5
  application cache
```

The Content Engine will listen on the ports listed in port list 5 (port 36) for redirected requests from the routers on router list 1.

Step 16  Enable the sixth user-defined WCCP service (service 95) on the Content Engine, and associate router list 1 and port list 6 with this service.

```
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 6
  application cache
```

The Content Engine will listen on the ports listed in port list 6 (port 37) for redirected requests from the routers on router list 1.

Step 17  Enable the seventh user-defined WCCP service (service 96) on the Content Engine, and associate router list 1 and port list 7 with this service.

```
ContentEngine(config)# wccp service-number 96 router-list-num 1 port-list-num 7
  application cache
```

The Content Engine will listen on the ports listed in port list 7 (port 38) for redirected requests from the routers on router list 1.

Step 18  Enable the eighth user-defined WCCP service (service 97) on the Content Engine, and associate router list 1 and port list 8 with this service.

```
ContentEngine(config)# wccp service-number 97 router-list-num 1 port-list-num 8
  application cache
```

The Content Engine will listen on the ports listed in port list 8 (port 39) for redirected requests from the routers on router list 1.

Step 19  Enable WCCP Version 2 on the Content Engine.

```
ContentEngine(config)# wccp version 2
```

Step 20  Disable the WCCP slow start feature on the Content Engine.

```
ContentEngine(config)# no wccp slow-start enable
```

For information about the WCCP slow start feature, see the “Configuring WCCP Slow Start” section on page 15-10.
Configuring Layer 4 Switching as a Redirection Method

To configure transparent redirection when Layer 4 switching (a Content Services Switch [CSS] switch) is being used to redirect requests transparently to a standalone Content Engine, keep these important points in mind:

- The CSS switch supports transparent proxy caching as well as reverse proxy caching. The CSS switch provides several load-balancing methods depending on how you want to distribute data over the Content Engines (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.
- 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 web server.
- If all cache servers are unavailable in a transparent cache configuration, the CSS switch allows all client requests to progress to the origin web servers.

To configure transparent caching with a CSS switch, complete the following tasks:

1. Enable transparent caching on the CSS switch.
2. Enable the standalone Content Engine to accept redirected traffic from the CSS switch.
3. Enable transparent caching on the standalone Content Engine.

The following sample workflow shows how to use CLI commands to configure transparent caching using a CSS switch and a standalone Content Engine. In this example, serv1 is configured as a transparent caching service using a CSS switch named CS150 and a standalone Content Engine named CE100. Ensure that you have configured interfaces, services, owners, VLANs and content rules prior to configuring caching with the CSS switch.

---

Note

Refer to the Content Services Switch Basic Configuration Guide for further information on how to configure these attributes on the CSS switch. For a complete description of each command, refer to the Content Services Switch Command Reference.

---

Example of Configuring Transparent Caching Using Layer 4 Switching

The following example shows how to enable transparent caching using a Layer 4 CSS switch (as opposed to a WCCP-enabled router) and a standalone Content Engine. To enable transparent caching, follow these steps:

---

Step 1  On the CSS switch, add service serv1 reserved for transparent caching.

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

Step 2  Specify transparent caching as the 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 switch 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 that you want the CSS switch to cache. Enter a text string containing from 1 to 8 characters.
CS150(config-eql[graphics]# extension jpeg
You can also provide a description of the extension type here. Enter a text string enclosed by quotation marks. The maximum length is 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 round-robin.
CS150(config-owner-content[cisco.com-rule1])# balance domain

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

Step 11 Exit configuration mode on the CSS switch.
CS150(config-owner-content[cisco.com-rule1])# end

Step 12 Save the configuration. The CSS switch is now configured for transparent caching services.
CS150(config-owner-content[cisco.com-rule1])# copy running-config startup-config

Step 13 Configure the standalone Content Engine to transparently receive Layer 4 redirected traffic from Layer 4-enabled switches such as the CSS switch.
CE100(config)# http 14-switch enable

Note  The http 14-switch command enables transparent redirection on HTTP port 80 only. If you want to intercept traffic on a different port, you must configure a WCCP service without a router, that contains the alternative port that you want to use.

If the version of ACNS that you are using does not accept the wccp command without the router-list num option, you can use a dummy router address. The Content Engine will accept redirected traffic and will send WCCP announcements to the configured router. You can avoid having the Content Engine send WCCP announcements to the configured router by later removing the dummy router list from the configured WCCP service; however, this configuration will be lost after you reload.
Step 14  Exit configuration mode on the standalone Content Engine.
CE100(config)# exit

Step 15  Write the running configuration to nonvolatile memory.
CE100# write memory
Configuring Conventional Caching Services for Standalone Content Engines

This chapter describes how to configure conventional caching services (HTTP, FTP [FTP-over-HTTP caching and native FTP caching], HTTPS, or DNS caching) for standalone Content Engines. It also describes how to configure the TFTP gateway, persistent connections, healing mode, and the Internet Cache Protocol (ICP) for standalone Content Engines. This chapter includes the following sections:

- Overview of Configuring Conventional Caching Services, page 7-2
- Configuring HTTP Caching for Standalone Content Engines, page 7-7
- Configuring HTTPS Caching for Standalone Content Engines, page 7-24
- Configuring FTP Caching for Standalone Content Engines, page 7-36
- Configuring the TFTP Server and Gateway for Standalone Content Engines, page 7-58
- Configuring DNS Caching for Standalone Content Engines, page 7-62
- Configuring Standalone Content Engines to Send out TCP Keepalives, page 7-67
- Configuring Persistent Connections on Standalone Content Engine, page 7-68
- Configuring Healing Mode for Content Engine Clusters, page 7-70
- Configuring the Internet Cache Protocol for Content Engine Clusters, page 7-72

Note

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication. For information about configuring caching for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Overview of Configuring Conventional Caching Services

This section provides an overview of how to use the Content Engine CLI to configure conventional caching services (HTTP, HTTPS, FTP, and DNS caching) on standalone Content Engines. Figure 7-1 provides a detailed view on how to configure conventional caching services for standalone Content Engines.

For information about how to use the Setup utility to configure the following three commonly used conventional caching services (HTTP reverse proxy caching, HTTP transparent caching using WCCP Version 2, and HTTP forward proxy caching), see the “Configuring a Basic Configuration on Standalone Content Engines with the Setup Utility” section on page 4-10.

For information about how to configure media caching services (WMT and RTSP caching and streaming services) in a locally managed deployment, see the following chapters:

- Chapter 8, “Configuring RealMedia Services on Standalone Content Engines”
- Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines”
Table 7-1 is a checklist of tasks for configuring conventional caching services (HTTP, HTTPS, FTP, and DNS caching) for standalone Content Engines. This checklist also includes the steps involved in configuring these services on a standalone Content Engine.
### Table 7-1 Checklist for Configuring Conventional Caching Services for Standalone Content Engines

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start basic configuration of conventional caching services</strong></td>
<td></td>
</tr>
</tbody>
</table>
| 1. Configure one or more of the following routing methods to direct client requests to the standalone Content Engine: | For direct proxy routing, see the “Configuring Client Browsers and Media Players for Direct Proxy Routing” section on page 4-35.  
For WCCP routing, see the “Configuring Standalone Content Engines for WCCP Transparent Redirection” section on page 6-9.  
For Layer 4 switching, see the “Configuring Layer 4 Switching as a Redirection Method” section on page 6-50. |
| – Direct proxy routing (nontransparent)                              |                                                                                                                                                                                                                                                                                                |
| – Transparent redirection (WCCP routing or Layer 4 switching)        |                                                                                                                                                                                                                                                                                                |
| 2. If direct proxy routing is to be used, is a *.pac file to be used? | • If no, then manually configure each client browser to point directly to the standalone Content Engine as a direct proxy server. See the “Manually Pointing Client Browsers to a Standalone Content Engine” section on page 4-42.  
• If yes, then configure the standalone Content Engine and the client browsers to use a proxy autoconfiguration (PAC) file. See the “Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine” section on page 4-37. |
| 3. Configure nontransparent (proxy) mode conventional caching services on this standalone Content Engine: | See the following sections in this chapter:  
• Configuring Nontransparent HTTP Forward Proxy Caching on Standalone Content Engines, page 7-8  
• Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines, page 7-38  
• Configuring HTTPS Proxy Caching for Standalone Content Engines, page 7-25 |
| – HTTP forward proxy caching                                          |                                                                                                                                                                                                                                                                                                |
| – FTP-over-HTTP caching                                              |                                                                                                                                                                                                                                                                                                |
| – HTTPS proxy caching                                                |                                                                                                                                                                                                                                                                                                |
| 4. Configure transparent mode conventional caching services for this standalone Content Engine: | See the following sections in this chapter:  
• Configuring HTTP Reverse Proxy Caching for Standalone Content Engines, page 7-23  
• Configuring FTP Native Caching for Standalone Content Engines, page 7-41  
• Configuring HTTPS Transparent Caching for Standalone Content Engines, page 7-27  
• Configuring Transparent HTTP Forward Proxy Caching for Standalone Content Engines, page 7-17  
• Configuring DNS Caching for Standalone Content Engines, page 7-62 |
| – HTTP reverse proxy caching                                          |                                                                                                                                                                                                                                                                                                |
| – Native FTP caching                                                 |                                                                                                                                                                                                                                                                                                |
| – HTTPS transparent caching                                          |                                                                                                                                                                                                                                                                                                |
| – Transparent HTTP forward proxy caching                            |                                                                                                                                                                                                                                                                                                |
| – DNS caching                                                        |                                                                                                                                                                                                                                                                                                |
Table 7-1 Checklist for Configuring Conventional Caching Services for Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>You can now do any of the following tasks: See tasks 6 through 24 below in this table.</td>
</tr>
<tr>
<td></td>
<td>– Configure the TFTP server and gateway.</td>
</tr>
<tr>
<td></td>
<td>– Configure the Content Engine to use persistent connections.</td>
</tr>
<tr>
<td></td>
<td>– Configure healing mode or ICP for cache clusters.</td>
</tr>
<tr>
<td></td>
<td>– Configure streaming services for the Content Engine.</td>
</tr>
<tr>
<td></td>
<td>– Configure content services for the Content Engine.</td>
</tr>
<tr>
<td></td>
<td>– Perform advanced configuration on the Content Engine.</td>
</tr>
<tr>
<td></td>
<td>– Monitor and troubleshoot.</td>
</tr>
<tr>
<td>6.</td>
<td>Configure the TFTP server and gateway on the Content Engine. Configuring the TFTP Server and Gateway for Standalone Content Engines, page 7-58</td>
</tr>
<tr>
<td>7.</td>
<td>Configure persistent connections for the Content Engine. Configuring Standalone Content Engines to Send out TCP Keepalives, page 7-67</td>
</tr>
<tr>
<td>8.</td>
<td>Configure healing mode for cache clusters. Configuring Healing Mode for Content Engine Clusters, page 7-70</td>
</tr>
<tr>
<td>10.</td>
<td>Configure WMT caching and streaming services on the Content Engine. Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines”</td>
</tr>
<tr>
<td>11.</td>
<td>Configure RTSP caching and streaming services on the Content Engine. Chapter 8, “Configuring RealMedia Services on Standalone Content Engines”</td>
</tr>
<tr>
<td></td>
<td>Configure content services (optional)</td>
</tr>
<tr>
<td>12.</td>
<td>After configuring caching and streaming services on the standalone Content Engine, you can configure such content services as access control, URL filtering, ICAP, and rules.</td>
</tr>
<tr>
<td>13.</td>
<td>Decide if end user access to the Internet is to be controlled (access control for HTTP, HTTPS, and FTP-over-HTTP requests).</td>
</tr>
<tr>
<td></td>
<td>• If no, then go to task 14.</td>
</tr>
<tr>
<td></td>
<td>• If yes, then configure content authentication and authorization, as described in Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.”</td>
</tr>
<tr>
<td>14.</td>
<td>Decide if URL filtering is to be used.</td>
</tr>
<tr>
<td></td>
<td>• If no, then go to task 14.</td>
</tr>
<tr>
<td></td>
<td>• If yes, then configure URL filtering for HTTP, HTTPS, and FTP requests, as described in Chapter 11, “Configuring Content Preloading and URL Filtering on Standalone Content Engines.”</td>
</tr>
</tbody>
</table>
## Overview of Configuring Conventional Caching Services

### Task 14. Determine whether there is an external Internet Content Adaptation Protocol (ICAP) server.
- If no, then go to task 15.
- If yes, then configure ICAP for HTTP and FTP-over-HTTP requests, as described in Chapter 12, “Configuring ICAP on Standalone Content Engines.”

### Task 15. Determine if there are any special requirements for processing content requests.
- If no, then go to task 16.
- If yes, configure rules for HTTP, HTTPS, FTP-over-HTTP, WMT, and RTSP requests, as described in Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”

## Perform advanced configuration tasks (optional)

### Task 16. Configure advanced transparent caching features (for example, traffic bypass, overload bypass, flow protection, and IP spoofing).
Chapter 15, “Configuring Advanced Transparent Caching Features on Standalone Content Engines”

### Task 17. Set up additional network interfaces on the standalone Content Engine.
Chapter 16, “Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines”

### Task 18. Configure bandwidth for interfaces and content services on this standalone Content Engine.
Chapter 16, “Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines”

### Task 19. Set up login authentication and authorization for administrative users who will be accessing the Content Engine for configuration, monitoring, or troubleshooting purposes.
Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines”

### Task 20. Configure this standalone Content Engine for system accounting with TACACS+.
Chapter 18, “Configuring AAA Accounting on Standalone Content Engines”

### Task 21. Configure IP access control lists (ACLs) on this standalone Content Engine.
Chapter 19, “Creating and Managing IP Access Control Lists for Standalone Content Engines”

### Task 22. View or modify TCP stack parameters for this standalone Content Engine.
Chapter 20, “Viewing and Modifying TCP Stack Parameters on Standalone Content Engines”

### Task 23. View or modify the system logging settings for this standalone Content Engine.
Monitoring the Performance of Specific URLs, page 21-52

## Monitor and troubleshoot

### Task 24. Monitor this standalone Content Engine with SNMP or the ACNS software alarms.
Chapter 21, “Monitoring Standalone Content Engines and Transactions”

### Task 25. Troubleshoot problems by using tools such as traceroute or ping.
Chapter 21, “Monitoring Standalone Content Engines and Transactions”

---

### Table 7-1 Checklist for Configuring Conventional Caching Services for Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
</table>
| 14.  | **Determine whether there is an external Internet Content Adaptation Protocol (ICAP) server.**  
- If no, then go to task 15.  
- If yes, then configure ICAP for HTTP and FTP-over-HTTP requests, as described in Chapter 12, “Configuring ICAP on Standalone Content Engines.” |
| 15.  | **Determine if there are any special requirements for processing content requests.**  
- If no, then go to task 16.  
- If yes, configure rules for HTTP, HTTPS, FTP-over-HTTP, WMT, and RTSP requests, as described in Chapter 13, “Configuring the Rules Template on Standalone Content Engines.” |
| 16.  | **Configure advanced transparent caching features (for example, traffic bypass, overload bypass, flow protection, and IP spoofing).**  
Chapter 15, “Configuring Advanced Transparent Caching Features on Standalone Content Engines” |
| 17.  | **Set up additional network interfaces on the standalone Content Engine.**  
Chapter 16, “Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines” |
| 18.  | **Configure bandwidth for interfaces and content services on this standalone Content Engine.**  
Chapter 16, “Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines” |
| 19.  | **Set up login authentication and authorization for administrative users who will be accessing the Content Engine for configuration, monitoring, or troubleshooting purposes.**  
Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines” |
| 20.  | **Configure this standalone Content Engine for system accounting with TACACS+.**  
Chapter 18, “Configuring AAA Accounting on Standalone Content Engines” |
| 21.  | **Configure IP access control lists (ACLs) on this standalone Content Engine.**  
Chapter 19, “Creating and Managing IP Access Control Lists for Standalone Content Engines” |
| 22.  | **View or modify TCP stack parameters for this standalone Content Engine.**  
Chapter 20, “Viewing and Modifying TCP Stack Parameters on Standalone Content Engines” |
| 23.  | **View or modify the system logging settings for this standalone Content Engine.**  
Monitoring the Performance of Specific URLs, page 21-52 |
| 24.  | **Monitor this standalone Content Engine with SNMP or the ACNS software alarms.**  
Chapter 21, “Monitoring Standalone Content Engines and Transactions” |
| 25.  | **Troubleshoot problems by using tools such as traceroute or ping.**  
Chapter 21, “Monitoring Standalone Content Engines and Transactions” |
Configuring HTTP Caching for Standalone Content Engines

HTTP is the main protocol used on the web for communication between web browsers and web servers. There are two commonly implemented HTTP versions today: HTTP 1.0 and HTTP 1.1. The ACNS 5.x software supports both HTTP 1.0 and HTTP 1.1.

HTTP runs over TCP port 80 (which is reserved for HTTP) and is a request-response protocol. The client (web browser) sends a request to a web server, and the web server responds with the content. Each request or response can carry a number of headers with it, specifying various properties of the client, the server, the object or communicating states between the client and the web server.

The HTTP 1.1 specification allows objects to be transmitted using Chunked Transfer Coding. In the ACNS 5.1 software and earlier releases, the proxying of chunked transfer encoded (CTE) objects was supported; however, the caching of these objects was not supported.

In the ACNS 5.2 software and later releases, caching of CTE HTTP objects is supported. A subsequent request for the same CTE object, which meets the standard HTTP caching freshness criteria, results in that object being fetched from the Content Engine’s cache and sent to the client that is HTTP 1.1 compliant. HTTP 1.0 clients do not support CTE. Consequently, if an object is stored in a CTE format, then the Content Engine must refetch the object from the source HTTP server if the HTTP client is not HTTP 1.1 compliant.

To enable the caching of CTE HTTP objects on a standalone Content Engine, specify the `http cache-chunk-encoded enable` global configuration command. After enabling this feature, you can use the `show statistics http request` EXEC command to verify that this feature is working properly. Check the command output to verify whether the displayed value in the Chunked HTTP Responses: field increments after the Content Engine serves cached CTE HTTP objects to HTTP 1.1 compliant clients.

Note
For details on HTTP, see the IETF RFC 1945 (HTTP 1.0 specification) and RFC 2616 (HTTP 1.1 specification).

Table 7-2 lists the HTTP caching services that are supported by Content Engines that are running the ACNS 5.2.1 software and later releases. The type of services supported varies based on the method used to route the HTTP request to the Content Engine.

<table>
<thead>
<tr>
<th>HTTP Caching Services</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Proxy Mode</td>
<td></td>
</tr>
<tr>
<td>Nontransparent HTTP forward proxy caching</td>
<td>Configuring Nontransparent HTTP Forward Proxy Caching on Standalone Content Engines, page 7-8</td>
</tr>
<tr>
<td>Transparent Redirection Mode</td>
<td></td>
</tr>
<tr>
<td>Transparent HTTP forward proxy caching</td>
<td>Configuring Transparent HTTP Forward Proxy Caching for Standalone Content Engines, page 7-17</td>
</tr>
<tr>
<td>Transparent HTTP reverse proxy caching</td>
<td>Configuring HTTP Reverse Proxy Caching for Standalone Content Engines, page 7-23</td>
</tr>
</tbody>
</table>

In the ACNS 5.1 software and earlier releases, a maximum of eight active WCCP services were supported by a WCCP Version 2 router and a Content Engine. In the ACNS 5.2.1 software and later releases, up to 25 active WCCP Version 2 services are supported. In ACNS 5.2.1 software release, only 17 WCCP services were defined. In the ACNS 5.3.1 software release, 18 WCCP services are defined.
Configuring Conventional Caching Services for Standalone Content Engines

See Table B-3 for a list of supported WCCP services.

Configuring Nontransparent HTTP Forward Proxy Caching on Standalone Content Engines

You can use the Content Engine GUI or CLI to configure HTTP proxy caching on a standalone Content Engine (nontransparent forward proxy server).

From the Content Engine GUI, choose Caching > HTTP Proxy, and use the displayed HTTP Proxy window to configure the HTTP connection settings. For more information on how to use this window, click the HELP button in the window.

From the Content Engine CLI, follow these steps:

**Step 1** Configure the client browsers to send their HTTP requests directly to the Content Engine (nontransparent forward proxy server).

Point the client browsers directly to the Content Engine so that HTTP requests from these browsers are sent directly to the Content Engine (direct proxy routing). You can use the proxy autoconfiguration feature (PAC file) or manually configure the client browsers by specifying the IP address and port number of the forward proxy server. For more information on this topic, see the “Configuring Client Browsers and Media Players for Direct Proxy Routing” section on page 4-35.

**Step 2** Configure the Content Engine to accept incoming HTTP requests on ports other than port 80.

```
ContentEngine(config)# http proxy incoming ports
```

*ports* is the port number used by the standalone Content Engine to receive HTTP requests directly from the client browsers. This number ranges from 1 to 65535. You can specify up to eight ports.

This example configures an incoming HTTP proxy on port 8080. Up to eight incoming proxy ports can be configured on the same command line.

```
ContentEngine(config)# http proxy incoming 8080
```

**Step 3** (Optional) Configure HTTP cache freshness for the Content Engine, as described in the “Configuring HTTP Cache Freshness Settings” section on page 7-9.

**Step 4** (Optional) Configure authenticated HTTP cache settings for the Content Engine, as described in the “Configuring Authenticated HTTP Cache Settings” section on page 7-12.
Configuring HTTP Cache Freshness Settings

With HTTP 1.1, you can configure the Content Engine to check the freshness of its cached objects before serving the requested content to a client browser. A fresh object is a web object that is not stale. A cached object is considered fresh under any of the following conditions:

- The Content Engine has freshly retrieved the object from the origin server.
- The Content Engine contacts the origin server to check about the freshness of the cached object, and the origin server confirms that the cached object has not been modified since the Content Engine cached it.
- The age of the cached object has not exceeded its freshness lifetime. The age of a cached object is the time that the object has been stored in the Content Engine’s cache without the Content Engine explicitly contacting the origin server to check if the object is still fresh.

You can use the If-Modified-Since (IMS) feature to configure a standalone Content Engine to revalidate the freshness of the content stored in its local cache before serving the content to a client browser. The Content Engine checks the freshness of its cached content under the following conditions:

- When the Content Engine receives an IMS message from the client browser. This occurs if the setting for the local cache on the client browser is configured to check for newer versions of the cached pages each time the page are accessed.
- When the Content Engine receives a request for expired content.

**Note**

If clients click their Reload browser button to reload the requested content into their browser, this causes all Content Engines that are located between the client and the origin servers that contain the requested content to refresh their cached objects.

The Content Engine validates the freshness of requested content in its cache by sending an IMS request to the origin web server. The Content Engine also sends an IMS request to the origin web server when the maximum Time To Live (TTL) has expired.

Content freshness is based upon a conditional GET feature of the HTTP protocol. The Content Engine will retrieve the requested information from the origin server again if the content has changed since it was cached on the Content Engine. In the HTTP protocol, the conditional GET request uses the value of the Last-Modified response header that was received with the document when it was retrieved and stored in the Content Engine cache. This value (the last modification date and time of the cached document) is sent in the If-Modified-Since request header. The conditional GET request uses the time stamp from the Last-Modified: header and sends it along with the request in the If-Modified-Since header.

The following example shows an IMS request that a Content Engine would send to an origin web server.

```
GET /index.html HTTP/1.1
Server: www.cisco.com
Connection: keep-alive
If-Modified-Since: Tue 12 Sep 2000 10:07:04 GMT   Accept: */*
```

If the content has not changed, the origin web server responds with a 304 Not Modified message, and does not send the content.

If the content has changed, the new version is transferred to the Content Engine again. Typically, the origin web server also sends the Content Engine a 200 OK response along with the new version of the content.
The following examples show these two possible responses from the origin web server to the Content Engine IMS request:

304 Not Modified
(end-of-request)

or

200 OK
(response headers)
(data)
(end-of-request)

The Expires response, which indicates the time that the response expires, also affects caching. This response header indicates the time that the response becomes stale and should not be sent to the client without an up-to-date check (using a conditional GET operation). If the HTTP header of a cached object does not specify an expiration time, the Content Engine can age out cached objects through the `http age-multiplier` and `http max-ttl` global configuration commands.

The Content Engine can also calculate an expiration time for each web object before it is written to disk. The Content Engine’s algorithm to calculate an object’s cache expiration date is as follows:

Expiration date = (Today’s date – Object’s last modified date) * Freshness factor

The last modified date is provided by the end server’s file system. The freshness factor is tunable and derived from the text and binary percentage parameters of the `http age-multiplier` global configuration command. Valid age-multiplier values are from 0 to 100 percent of the object’s age. Default values are 30 percent for text (HTML) and 60 percent for binary objects (for example, gifs). After the expiration date has passed, the object is considered stale and subsequent requests causes a fresh retrieval of the content by the Content Engine.

When configuring the HTTP cache freshness settings on standalone Content Engines, keep the following important points in mind:

- You can specify the maximum size of an HTTP object that can be stored in the cache. The maximum size limit for an HTTP object is 2096128 kilobytes (2 GB). An object with a size above the configurable upper limit is not stored by the Content Engine.

- Use the minimum and maximum Time To Live (TTL) settings to limit the duration of HTTP 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 has an explicit expiration date, this takes precedence over the configurable TTL. The default values are 3 days for text files and 7 days for binary objects.

- For HTTP objects, use the `http min-ttl` and `ftp min-ttl` global configuration commands to set the minimum TTL.

- Use the `http cache-cookies` global configuration command to enable the Content Engine to cache binary objects that are served with HTTP set-cookies headers and no explicit expiration information, but which might be cacheable.
You can use the Content Engine CLI or GUI to configure the freshness settings for an HTTP cache on a standalone Content Engine.

From the Content Engine GUI, choose **Cache > HTTP Freshness**, and use the displayed HTTP Freshness window. To obtain more information about how to use the HTTP Freshness window to configure freshness settings, click the **HELP** button in the window.

From the Content Engine CLI, follow these steps:

**Step 1** Specify the freshness factor for HTTP cached objects:

```
ContentEngine(config)# http age-multiplier text 50% bin 70%
```

**Step 2** Set the minimum amount of time that the HTTP cached object is stored in the Content Engine cache. In the following example, this minimum time is set to 10 minutes:

```
ContentEngine(config)# http min-ttl 10
```

**Step 3** Set the upper limit on the estimated expiration dates for HTTP cached objects, as indicated in the following examples:

```
ContentEngine(config)# http max-ttl days text 2 binary 4
ContentEngine(config)# http max-ttl hours text 1 hours binary 4
```

The TTL sets a ceiling on estimated expiration dates. An explicit expiration date in the HTTP header takes precedence over the configured TTL. Table 7-3 lists the valid range of values.

**Table 7-3** Time To Live Range of Values for HTTP Freshness

<table>
<thead>
<tr>
<th>Scale</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Days</td>
<td>1–1825</td>
</tr>
<tr>
<td>Hours</td>
<td>1–43800</td>
</tr>
<tr>
<td>Minutes</td>
<td>1–2628000</td>
</tr>
<tr>
<td>Seconds</td>
<td>1–157680000</td>
</tr>
</tbody>
</table>

**Step 4** Specify the method that the Content Engine is to use to handle requests to revalidate the content freshness of the HTTP objects in its cache. In the following example, the Content Engine is configured to revalidate all HTTP objects for every HTTP request:

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

**Step 5** Set the upper limit of the HTTP object size in kilobytes (KB). In the following example, the maximum size for an HTTP object is set to 500 kilobytes:

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

**Note** The Content Engine does not store an object if the size of the object exceeds the specified limit. The maximum object size for cached HTTP objects is 2096128 KB (2 GB).

**Step 6** Configure the Content Engine to cache binary objects and associated cookies, which are served with HTTP set-cookies headers and no explicit expiration information, but which might be cacheable:

```
ContentEngine(config)# http cache-cookies
```
### Configuring Authenticated HTTP Cache Settings

The authenticated HTTP caching feature allows content that was authenticated through basic authentication and NTLM authentication 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.

**Note**

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.

You can use the Content Engine GUI or CLI to configure the parameters for the HTTP authentication cache on standalone Content Engines:

- From the Content Engine GUI, choose **Caching > Auth.Cache**, and use the displayed Authenticated Cache window. To obtain more information about this window, click the **HELP** button in the window.

- From the Content Engine CLI, use the `http authentication` global configuration command, as described in Table 7-4.

```plaintext
http authentication { cache { max-entries entries | max-group-entries number | timeout minutes | ttl minutes } | header { 401 | 407 } | realm line }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>Configures parameters that are related to the authentication cache on the Content Engine.</td>
</tr>
<tr>
<td>max-entries entries</td>
<td>Sets the maximum number of entries retained in the authentication cache.</td>
</tr>
<tr>
<td>max-group-entries number</td>
<td>Maximum number of entries retained in the authentication cache on the Content Engine. Valid values are from 500 to 32000 entries. This is limited by the physical resources available on the Content Engine.</td>
</tr>
<tr>
<td>realm line</td>
<td>Sets the maximum number of entries retained in the authentication group cache. This is subject to physical resources on the Content Engine. This option is only available in the ACNS 5.2 software and later releases.</td>
</tr>
</tbody>
</table>

**Tip**

If you intend to use the group name pattern, make sure that you set the correct number of maximum group entries in the authentication group cache. This number should correspond to the maximum number of groups that could be returned during authorization queries (for example, the total number of groups defined on the AAA server.)
Table 7-4 Parameters for the http authentication CLI Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Maximum number of entries retained in the authentication group cache on the Content Engine. Valid values are from 500 to 12000 entries. This is limited by the physical resources available on the Content Engine.</td>
</tr>
<tr>
<td>timeout</td>
<td>Sets the timeout value of records in the authentication cache. 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 a server lookup for reauthentication. This is the least-recently-used (LRU) value. It is also referred to as the idle time.</td>
</tr>
<tr>
<td>minutes</td>
<td>Time in minutes (1–1440) between the user’s last Internet access and the removal of that user’s entry from the authorization cache, forcing reauthentication. The default is 240 minutes (4 hours); the minimum is 30 minutes; and the maximum is 1440 minutes (24 hours).</td>
</tr>
<tr>
<td>ttl</td>
<td>Sets an absolute Time To Live (TTL) for entries in the authentication cache. This option is only available in the ACNS 5.2.1 software and later releases. By default, this option is disabled, which means that there is no TTL timeout in effect. This means that there will be no check to time out an authentication cache entry based on its creation time relative to a TTL value.</td>
</tr>
<tr>
<td>minutes</td>
<td>Time in minutes (1–1440) that specifies the maximum amount of time that an entry is valid in the authentication cache entry after its creation. The minimum is 1 minute; the maximum is 1440 minutes (24 hours). For more information, see the “Specifying a Reauthentication Interval” section on page 7-14.</td>
</tr>
<tr>
<td>header</td>
<td>Determines which HTTP header to use for authentication (user ID and password) when the style of the HTTP request indicates that no proxy server is present. Headers can be either HTTP 401 (Unauthorized) or HTTP 407 (Proxy Authentication Required). The default is HTTP 401.</td>
</tr>
<tr>
<td>401</td>
<td>Uses HTTP 401 to query users for credentials.</td>
</tr>
<tr>
<td>407</td>
<td>Uses HTTP 407 to query users for credentials.</td>
</tr>
<tr>
<td>realm</td>
<td>Configures the realm string for basic HTTP request authentication.</td>
</tr>
<tr>
<td>line</td>
<td>Name of the realm string to be authenticated. The default is Cisco Content Engine.</td>
</tr>
</tbody>
</table>

The maximum number of entries that is maintained in the authentication cache is 32,000. The minimum number is 500. The default value is 16,000 entries. Use the http authentication max-entries global configuration command to configure the maximum number of entries that is to be maintained in the authentication cache on this Content Engine, if necessary.

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 default time interval between the user’s last Internet access and the removal of that user’s entry from the authorization cache is 240 minutes (4 hours). The minimum time interval is 1 minute, and the maximum is 1440 minutes (24 hours). The Content Engine forces reauthentication with the access control server once this time interval expires.

In this example, the length of time that entries are valid in the authentication cache is set at 1000 minutes:

```
ContentEngine(config) # http authentication cache timeout 1000
```
Configuring HTTP Caching for Standalone Content Engines

When LDAP, RADIUS, and TACACS+ are used in proxy redirection mode, the authentication record kept in the authentication cache is indexed by the username and the password entered. When LDAP, RADIUS, or TACACS+ is used in WCCP-enabled router redirection mode, the authentication record indexed is the IP address of the Content Engine that is sending the request in transparent mode. When an NTLM server is used in either proxy redirection mode or WCCP-enabled router redirection mode, all authentication records are indexed by using the IP address of the requesting client. By default, the Content Engine authenticates cache loads based on the URL syntax of the incoming request.

Use the `http authentication header` global configuration command to configure the Content Engine to send a message to the client when authorization has failed. You can choose `http authentication header 401` (Unauthorized) or `http authentication header 407` (Proxy Authorization Required).

In the following example, the Content Engine is configured to use the 407 header when asking end users for their authentication credentials (user ID and password):

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

**Note**

In the ACNS 5.2.1 software and later releases, the ability to send HTTP transaction log messages to a remote syslog server is supported. This allows you to monitor the remote syslog server for HTTP request authentication failures in real time. For more information, see the “Monitoring HTTP Request Authentication Failures in Real Time” section on page 21-48.

**Specifying a Reauthentication Interval**

In the ACNS 5.1 software, an inactivity timer was used to determine when the client browser was prompted to reenter authentication credentials after being initially authenticated. This inactivity timer is configured with the `http authentication cache timeout` global configuration command. By default, the inactivity timeout period was 8 hours. This meant that as long as someone continued to use the client browser after the legitimate authenticated user was initially authenticated, the client was not forced to reenter that person’s authentication credentials.

In the ACNS 5.2.1 software and later releases, you can specify an absolute TTL for HTTP authentication cache entries for increased security in a shared workstation environment. This ability to specify an absolute TTL timeout adds security to the inactivity timeout mechanism for content that is served through the Content Engine. When the absolute TTL period has expired, the client browser is forced to reauthenticate itself, and the user must enter valid credentials.

A security vulnerability exists in a shared workstation environment that is using WCCP-enabled router redirection mode with any authentication method, or proxy redirection mode and the NTLM authentication method. In these cases, the Content Engine uses the client’s IP address as the index into the authentication record kept in the authentication cache, and the Content Engine can therefore authenticate users who have not presented valid credentials of their own if a different user using the same workstation has previously presented valid credentials that are cached in the authentication cache. To provide additional security in this situation, you can configure the absolute TTL for HTTP authentication cache entries; this will specify the absolute time during which an authentication cache entry is valid in the cache. If a cache lookup occurs on an entry and its configured TTL time is exceeded, then the entry is deleted and the Content Engine will query the user for credentials.
To support this feature, the `ttl` option was added to the `http authentication cache` global configuration command.

```
ContentEngine(config)# http authentication cache?
max-entries        Maximum entries in authentication cache; this is subject to
physical resources on the platform
max-group-entries   Maximum entries in authentication group cache; this is
subject to physical resources on the platform
timeout            Amount of time between last access and cache removal
ttl                Maximum amount of time from creation an entry is valid in
                   the cache
```

In order to use the absolute TTL effectively in a shared workstation environment that uses the Internet Explorer browser, there are additional considerations, because by default the browser will automatically send the workstation logon credentials when the Content Engine queries the user for credentials. Therefore, for the absolute TTL to provide additional security, either the logon credentials must not be valid request authentication credentials for the Content Engine or the Internet Explorer browser must configure its security settings to not send the logon credentials automatically when the Content Engine queries the user for credentials. To configure this on the Internet Explorer browser, choose `Tools > Internet Options > Security > Internet (and/or the other zones) > Custom Level > User Authentication > Logon > Prompt for Username and Password`. For this browser configuration to be effective, it must receive a 401 HTTP reply code when it is queried for credentials, which is the default in transparent redirection using WCCP. To have the 401 HTTP reply code used when in proxy redirection mode, use the Content Engine `http authentication header 401` global configuration command.

---

**Note**

In a shared workstation environment, the assumption is that each user will close the browser before leaving the shared workstation. If a user does not close the browser session, other users can continue to use that browser session without being asked to enter their own username and password credentials because the browser automatically sends the credentials for a session that has already been authenticated with the proxy.

---

For non-shared workstations that use the Internet Explorer browser, configure the Internet Explorer browsers on these workstations to automatically send their logon credentials by choosing `Tools > Internet Options > Security > Internet (and/or the other zones) > Custom Level > User Authentication > Logon > Automatic logon only in Intranet zone (or Automatic logon with current username and password)`. The workstation logon credentials of the nonshared workstation users must be the same credentials that will successfully authenticate with the Content Engine. This explicit configuration of the browser is only needed if WCCP mode is being used. In proxy mode, the client browsers will always send the credentials.

This absolute TTL timeout (the `ttl` option) allows you to specify an absolute value for the maximum time that an authentication cache entry is considered valid. The TTL timeout is specified in minutes (1–1440). The minimum is 1 minute; the maximum is 1440 minutes (24 hours). By default, this absolute TTL timeout option is disabled on the Content Engine.

Although a short absolute configuration value increases security in a shared workstation environment, it also increases the number of requests that must be made to the authentication server.

---

**Note**

The existing inactivity timeout (the `timeout` option) is not affected by the absolute TTL (the `ttl` option); they are independent of each other. If both timeouts (the inactivity timeout and TTL timeout) are configured on the Content Engine, the authentication cache entry times out because either the inactivity timeout or the TTL timeout occurred depending on which timeout occurs first for that entry.
Displaying the Configuration of the HTTP Authentication Cache

To display the current configuration of the HTTP authentication cache on a standalone Content Engine, enter the `show http authentication` EXEC command.

```
ContentEngine# show http authentication
HTTP Authentication:
   Header: Based on URL syntax
   Realm: "Cisco Content Engine"
   Cache Timeout: 480 (minutes)
   Cache TTL: 240 (minutes)
   Cache Maximum entries configured: 8000
   Cache Maximum entries platform limit: 8000
   Group Cache Maximum entries configured: 500
   Group Cache Maximum entries platform limit: 1000
```

Enabling the Caching of Authenticated Content

By default, authenticated content is not cached in HTTP. To change this default policy, use the `http cache-authenticated` global configuration command.

```
http cache-authenticated {all | basic | ntlm}
```

Table 7-5 describes these command parameters.

### Table 7-5 Parameters for the http cache-authenticated CLI Command

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cache-authenticated</code></td>
<td>Caches and revalidates authenticated web objects.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>Caches web objects that were authenticated with any authentication scheme</td>
</tr>
<tr>
<td><code>basic</code></td>
<td>Caches web objects that were authenticated with the basic authentication</td>
</tr>
<tr>
<td><code>ntlm</code></td>
<td>Caches web objects that were authenticated with the NTLM authentication</td>
</tr>
</tbody>
</table>

Enable caching of both basic and NTLM authenticated content on the Content Engine:

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

Verify which types of object caching are currently enabled on the Content Engine:

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

Enable NTLM object caching on a standalone Content Engine:

```
ContentEngine(config)# http cache-authenticated ntlm
```

When NTLM object caching is enabled on a Content Engine, when the Content Engine receives an NTLM HTTP request from a client, it searches its cache for the requested content. If a cache hit occurs, the Content Engines sends the client the requested content. If a cache miss occurs, the Content Engine retrieves the requested content from the origin server, caches a local copy of the content, and sends the requested content to the client.
The cached objects are tagged as NTLM protected so that subsequent requests for these same objects are subjected to authentication before the Content Engine can serve the content to the client. If there is a cache hit and the requested object is an NTLM protected object, the Content Engine checks whether there is a secured connection between this client and the server.

- If there is, the Content Engine sends an if-modified-since (IMS) request to the server using the proxy server connection.
- If the Content Engine receives a 304 response from the server, the Content Engine serves the cached content to the client. If the Content Engine receives a 200 response from the server, the Content Engine caches the new object and serves it to the client. If there is no established secure connection between the client and the server, the Content Engine attempts to establish the secure connection using IMS messages.

### Displaying HTTP Authentication Cache Statistics

To display authentication cache statistics for a standalone Content Engine, use the `show statistics http-authcache` EXEC command:

- **Dels TTL** shows the number of entries that were deleted from the authentication cache because of the absolute TTL timeout (the `http authentication cache ttl` command option).
- **DelTimeout** shows the number of entries that were deleted from the authentication cache because of the inactivity timer (the `http authentication cache timeout` command option).
- **Dels Other** shows the number of entries deleted from the authentication cache for all other reasons (for example, entries that were deleted because the `clear users request-authenticated` EXEC command was entered).

### Configuring Transparent HTTP Forward Proxy Caching for Standalone Content Engines

When transparent redirection is being used to direct HTTP requests to a Content Engine, you can configure the following caching services on the Content Engine:

- The standard web-cache service (service 0), as described in the “Configuring the Standard Web-Cache Service (Service 0) for Standalone Content Engines” section on page 7-18.
- The custom-web cache service (service 98), as described in the “Configuring the Custom Web-Cache Service (Service 98) for Standalone Content Engines” section on page 7-20.

You can use the Content Engine GUI or CLI to configure these caching services on a standalone Content Engine. If you use the Content Engine GUI to enable and configure the standard web-cache service (service 0) or the custom-web-cache service (service 98) on a standalone Content Engine, then you must specify the designated router list for each of these WCCP services in the following Content Engine GUI windows: the Web Cache window (WCCP > Web Cache), and the Custom Web Cache window (WCCP > Custom Web Cache). For more information on how to use the Custom Web Cache window, click the **HELP** button in the window.
Chapter 7  Configuring Conventional Caching Services for Standalone Content Engines

Configuring the Standard Web-Cache Service (Service 0) for Standalone Content Engines

The standard web-cache service (service 0) permits a single WCCP Version 1-enabled router or one or more WCCP Version 2-enabled routers to redirect HTTP traffic to standalone Content Engines on port 80 only. In order for a standalone Content Engine to accept redirected HTTP requests on port 80, you must configure the standard web-cache service on the Content Engine (transparent HTTP forward proxy caching).

To configure a standalone Content Engine to run the web-cache service (service 0) with WCCP Version 2, use the `wccp web-cache` global configuration command. To disable this service, use the `no` form of this command.

```
wccp web-cache {mask {[dst-ip-mask hex_num] [dst-port-mask port_hex_num] [src-ip-mask hex_num] [src-port-mask port_hex_num]} | router-list-num num [assign-method-strict] [l2-redirect] [mask-assign] [password key] [weight percentage]}
```

Table 7-6 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>Sets the mask used for Content Engine assignment. Configure at least 1; the maximum is 4.</td>
</tr>
<tr>
<td>dst-ip-mask</td>
<td>(Optional) Sets the mask used to match the packet destination IP address.</td>
</tr>
<tr>
<td>hex_num</td>
<td>IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000–FC000000.</td>
</tr>
<tr>
<td>dst-port-mask</td>
<td>(Optional) Sets the mask used to match the packet destination port number.</td>
</tr>
<tr>
<td>port_hex_num</td>
<td>Destination port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0–65535.</td>
</tr>
<tr>
<td>src-ip-mask</td>
<td>(Optional) Sets the mask used to match the packet source IP address.</td>
</tr>
<tr>
<td>src-port-mask</td>
<td>(Optional) Sets the mask used to match the packet source port number.</td>
</tr>
<tr>
<td>router-list-num</td>
<td>Sets the router list number.</td>
</tr>
<tr>
<td>num</td>
<td>Router list number (1–8).</td>
</tr>
<tr>
<td>assign-method-strict</td>
<td>Forces WCCP to strictly use only the configured assignment method.</td>
</tr>
<tr>
<td>l2-redirect</td>
<td>(Optional) Packet forwarding by Layer 2 redirect.</td>
</tr>
<tr>
<td>mask-assign</td>
<td>(Optional) Uses the mask method for Content Engine assignment.</td>
</tr>
<tr>
<td>password</td>
<td>(Optional) Sets the authentication password.</td>
</tr>
<tr>
<td>key</td>
<td>WCCP service password key.</td>
</tr>
<tr>
<td>weight</td>
<td>(Optional) Sets weight percentage for load balancing.</td>
</tr>
<tr>
<td>percentage</td>
<td>Percentage value (0–100).</td>
</tr>
</tbody>
</table>

In the following example, the standard web-cache service (service 0) is configured on a standalone Content Engine and a WCCP-enabled router. By configuring this WCCP service on the router, the WCCP router redirects HTTP requests transparently to the Content Engine on port 80. By configuring this service on the Content Engine, the Content Engine listens on port 80 for redirected HTTP requests. If the Content Engine determines that it should accept and process the redirected HTTP request, it
retrieves the requested information from the origin server if it is not already stored in its cache, caches a copy of the content in its local storage if the content is cacheable, and then send the requested content to the client browser.

To configure the standard web-cache service (service 0) on a Content Engine and a WCCP router, follow these steps:

---

**Step 1** Enable WCCP Version 1 or Version 2 on the standalone Content Engine:

ContentEngine(config)# wccp version 1

or

ContentEngine(config)# wccp version 2

The Content Engine must be running WCCP Version 2 to support any of the WCCP services listed in Table B-3 other than the web-cache service (service 0). If you enable WCCP Version 1 as opposed to Version 2 on this Content Engine, only a single WCCP router can be configured to support the only supported service (the standard web-cache service). If you select Version 2, up to eight WCCP routers can be specified to support a particular WCCP service, and all WCCP services are supported.

---

**Tip** You can also enable WCCP Version 1 or Version 2 on a standalone Content Engine by choosing WCCP > Enable WCCP from the Content Engine GUI and then clicking the Version 1 or Version 2 radio button.

---

**Step 2** Create a router list that specifies the routers that will support the web cache service. Enter the IP address or multicast address of every router that will support the web-cache service for this Content Engine. If different routers will be used for different WCCP services, you must create more than one router list. In this case, there is only one router on router list 1 (the router that you just configured for the standard web-cache service, which has an IP address of 10.0.1.1):

ContentEngine(config)# wccp router-list 1 10.0.1.1

If you use the Content Engine GUI to enable and configure WCCP on this Content Engine, then you must specify the designated router list for the web-cache service in the Web Cache window (WCCP > Web Cache) of the Content Engine GUI.

**Step 3** Inform the WCCP-enabled router in the specified router list that this standalone Content Engine is accepting redirected web cache requests on port 80:

ContentEngine(config)# wccp web-cache router-list-num 1

**Step 4** Exit global configuration mode.

ContentEngine(config)# exit

**Step 5** Write the running configurations to nonvolatile memory.

ContentEngine# write memory

**Step 6** Enable WCCP Version 2 on the router.

Router# configure terminal
Router(config)# ip wccp version 2

**Step 7** Enable the standard web-cache service on the router.

Router(config)# ip wccp web-cache
Step 8  Specify the interface on which the standard web-cache service will run.

Router(config)# interface type number

In the following example, Ethernet interface 0/1 on the router is configured to run the standard web-cache service:

Router(config)# interface ethernet 0/1

Step 9  Configure the router to check the HTTP traffic that arrives on the interface on which the standard web-cache service is configured (for example, Ethernet interface 0/1). The router will determine whether it should redirect these packets to the standalone Content Engine. This Content Engine functions as a transparent forward proxy server that will accept redirected HTTP requests on port 80 from this WCCP Version 2 router.

Router(config-if)# ip wccp web-cache redirect in

Configuring the Custom Web-Cache Service (Service 98) for Standalone Content Engines

To enable a standalone Content Engine to accept redirected HTTP traffic on a port other than 80, configure the Content Engine to support the custom-web-cache service (service 98).

The `wccp custom-web-cache` global configuration command causes the Content Engine to establish WCCP Version 2 redirection services automatically with a Cisco router on a user-specified port number. The Content Engine then performs transparent web caching for all HTTP requests over that port while port 80 transparent web caching continues without interruption. For custom web caching, the custom-web-cache service (service 98) must be enabled on routers that are running WCCP Version 2. WCCP Version 1 does not support the custom-web-cache service.

To configure the custom-web-cache service on a standalone Content Engine, use the `wccp custom-web-cache` global configuration command. To disable the custom-web-cache service (service 98) on the Content Engine, use the `no` form of this command.

`wccp custom-web-cache [mask {[dst-ip-mask hex_num] [dst-port-mask port_hex_num] [src-ip-mask hex_num] [src-port-mask port_hex_num]} | router-list-num num port port [assign-method-strict] [hash-destination-ip] [hash-destination-port] [hash-source-ip] [hash-source-port] [l2-redirect] [mask-assign] [password key] [weight percentage]}

Table 7-7 describes these command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>Sets the mask used for Content Engine assignment. Configure at least 1 and up to 4.</td>
</tr>
<tr>
<td>dst-ip-mask</td>
<td>(Optional) Sets the mask used to match the packet destination IP address.</td>
</tr>
<tr>
<td>hex_num</td>
<td>IP address mask defined by a hexadecimal number (for example, 0xFC0000000).</td>
</tr>
<tr>
<td></td>
<td>The range is 0x00000000–FC0000000.</td>
</tr>
<tr>
<td>dst-port-mask</td>
<td>(Optional) Sets the mask used to match the packet destination port number.</td>
</tr>
<tr>
<td>port_hex_num</td>
<td>Port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0–65535.</td>
</tr>
</tbody>
</table>
The `l2-redirect` option permits the Content Engine to receive transparently redirected traffic from a WCCP Version 2-enabled router or switch if the Content Engine has a Layer 2 connection with the device and the device is configured for Layer 2 redirection.

The `weight` parameter represents a percentage of load redirected to the Content Engine cluster (for example, a Content Engine with a weight of 30 receives 30 percent of the total load). If the total of all weight parameters in the Content Engine cluster exceeds 100, the percentage load for each Content Engine is recalculated as the percentage that its weight parameter represents of the combined total.

In the following example, a standalone Content Engine and a router that are both running WCCP Version 2 are configured to support the custom-web-cache service (service 98). By configuring this WCCP caching service, the WCCP Version 2 routers can redirect HTTP traffic transparently to a Content Engine (transparent forward proxy server) on multiple ports (up to eight ports). By configuring the custom-web-cache service on the Content Engine and router, the Content Engine can intercept HTTP requests on multiple ports without having to configure a user-defined WCCP service (services 90 to 97). For more information about configuring user-defined WCCP services on a Content Engine and a router, respectively, see the “Configuring WCCP Services on Standalone Content Engines” section on page 6-14 and the “Configuring WCCP Services on a Router” section on page 6-27.
To configure the custom-web-cache service (service 98) on a standalone Content Engine and a router, follow these steps:

**Step 1**  
Enable WCCP Version 2 on the standalone Content Engine:  
`ContentEngine(config)# wccp version 2`

The Content Engine must be running WCCP Version 2 to support the custom-web-cache service (service 98). WCCP Version 2 is required for any of the WCCP services listed in Table B-3 other than the standard web-cache service (service 0).

**Tip**  
You can also enable WCCP Version 2 on a standalone Content Engine by choosing **WCCP > Enable WCCP** from the Content Engine GUI and then clicking the **Version 1** or **Version 2** radio buttons.

**Step 2**  
Create a router list that specifies the routers that will support the custom-web-cache service (service 98). Enter the IP address or multicast address of every router that will support the custom-web-cache service for this Content Engine. If different routers will be used for different WCCP services, you must create more than one router list.

In this case, there is only one router on router list 1 (the router that you just configured for the custom-web-cache service, which has an IP address of 10.0.1.1).  
`ContentEngine(config)# wccp router-list 1 10.0.1.1`

If you use the Content Engine GUI to enable and configure WCCP on this Content Engine, then you must specify the designated router list for the custom-web-cache service in the Custom Web Cache window (**WCCP > Custom Web Cache**) of the Content Engine GUI.

**Step 3**  
Inform the WCCP-enabled router in the specified router list that this standalone Content Engine is accepting redirected custom web cache requests on port 31.  
`ContentEngine(config)# wccp custom-web-cache router-list-num 1 port 31`

**Step 4**  
Exit global configuration mode.  
`ContentEngine(config)# exit`

**Step 5**  
Write the running configurations to nonvolatile memory.  
`ContentEngine# write memory`

**Step 6**  
Enable WCCP Version 2 on the router.  
`Router# configure terminal`  
`Router(config)# ip wccp version 2`

**Step 7**  
Enable the custom-web-cache service (service 98) on the router.  
`Router(config)# ip wccp 98`

**Step 8**  
Specify the interface on which the custom-web-cache service will run.  
`Router(config)# interface type number`

In the following example, the Ethernet 0 interface on the router is configured to run the custom-web-cache service:  
`Router(config)# interface ethernet 0`
Step 9 Configure the router to use the outbound interface for the custom-web-cache service.

Router(config-if)# ip wccp 98 redirect out

---

Configuring HTTP Reverse Proxy Caching for Standalone Content Engines

The reverse-proxy service (service 99) is the predefined WCCP Version 2 service that permits WCCP Version 2-enabled routers to redirect reverse proxy packets to a standalone Content Engine that is functioning as a transparent reverse proxy server.

This section provides an example of how to configure this WCCP service with a WCCP Version 2-enabled router and a standalone Content Engine. In this example, WCCP Version 2 and the reverse-proxy service (service 99) are used to redirect HTTP reverse proxy requests transparently to the Content Engine on port 80. By configuring this service on the Content Engine, the Content Engine processes the redirected HTTP reverse proxy requests. As part of processing the request, the Content Engine retrieves the requested information from the origin server if it is not already stored in its cache, caches a local copy, and sends the requested content to client browser.

To configure the reverse-proxy service (service 99) on a standalone Content Engine and a WCCP router, follow these steps:

Step 1 Enable WCCP Version 2 on the standalone Content Engine.

ContentEngine(config)# wccp version 2

The Content Engine must be running WCCP Version 2 to support the reverse-proxy service. WCCP Version 1 only supports the standard web-cache service (service 0); it does not support the reverse-proxy service (service 99). See Table B-3 for a list of the WCCP services.

Tip You can also enable WCCP Version 2 on a standalone Content Engine by choosing WCCP > Enable WCCP from the Content Engine GUI and then clicking the Version 2 radio buttons.

Step 2 Create a router list that specifies the routers that will support the reverse-proxy service. Enter the IP address or multicast address of every router that will support this WCCP service. If different routers will be used for different WCCP services, you must create more than one router list.

In this example, there is only one router on router list 1 (the router that you just configured for the reverse proxy cache service, which has an IP address of 10.0.1.1):

ContentEngine(config)# wccp router-list 1 10.0.1.1

If you use the Content Engine GUI to enable and configure WCCP on this Content Engine, then you must specify the designated router list for the reverse-proxy service in the Reverse Proxy window (WCCP > Reverse Proxy).

Step 3 Inform the WCCP-enabled router in the specified router list that this standalone Content Engine is accepting redirected reverse proxy cache requests on port 80.

ContentEngine(config)# wccp reverse-proxy router-list-num 1

Step 4 Exit global configuration mode.

ContentEngine(config)# exit
Step 5  Write the running configurations to nonvolatile memory.

ContentEngine# write memory

Step 6  Enable WCCP Version 2 on the router.

Router# configure terminal
Router(config)# ip wccp version 2

Step 7  Enable the reverse-proxy service (service 99) on the router.

Router(config)# ip wccp 99

Step 8  Specify the interface on which the reverse-proxy service will run.

Router(config)# interface type number

In the following example, the Ethernet 0 interface on the router is configured to run the reverse-proxy service:

Router(config)# interface ethernet 0

Configure the router to use the outbound interface for the reverse-proxy service. The router will check the reverse proxy packets on Ethernet interface 0 to determine if it should transparently redirect these packets to the Content Engine (transparent reverse proxy server).

Router(config-if)# ip wccp 99 redirect out

---

**Configuring HTTPS Caching for Standalone Content Engines**

The HTTPS protocol is essentially the HTTP protocol running over a Secure Socket Layer (SSL) transport. SSL is a protocol that provides a secure channel between two devices (for example, a client and a server). SSL uses public-key cryptography to ensure the security and privacy of the exchange between the client browser and the server. HTTPS uses a unique URL that begins with https:// (for example, https://abc.com). The default port number for HTTPS is port 443 instead of port 80, which is the default port for HTTP.

Content Engines can either SSL terminate or tunnel HTTPS client requests to the origin HTTPS server, as described in the following sections:

- **About SSL Termination of HTTPS Client Requests, page 7-25**
- **About Tunneling of HTTPS Client Requests, page 7-25**

For information about how to configure the different types of HTTPS caching on a standalone Content Engine, see:

- **Configuring HTTPS Proxy Caching for Standalone Content Engines, page 7-25**
- **Configuring HTTPS Transparent Caching for Standalone Content Engines, page 7-27**
About SSL Termination of HTTPS Client Requests

If the Content Engine SSL terminates an HTTPS client request, this means that it decrypts the SSL-encrypted data. By decrypting the SSL-encrypted data, the Content Engine can see the HTTPS client request in plain text, which allows the Content Engine to perform numerous HTTP processing tasks (for example, caching, rule processing, and filtering) on such requests. Content Engines running the ACNS 5.1 software and later releases, support this SSL-termiation feature. Content Engines running the ACNS 5.2 software and later releases, can also rewrite and redirect HTTPS requests as defined by the specified rules. If the Content Engine terminates an HTTPS request, it can apply most of the rules on the HTTPS request. (The no-proxy, use-icap-service, and use-proxy rule actions are not supported for HTTPS caching.) For more information, see Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”

For the SSL-termination feature to work properly, you must install the SSL certificate and private key of the origin HTTPS servers on the Content Engine. The SSL-termination feature works in both transparent mode and proxy mode if the Content Engine has the correct certificates and private keys installed. For specific requested content to be cached, you must import the proper certificates and keys for these origin HTTPS servers into the Content Engine and configure the Content Engine to cache content from these origin HTTPS servers. For standalone Content Engines, this is performed through the Content Engine CLI, as described in the “Configuring Certificates and Private Keys for HTTPS Caching” section on page 7-32.

In the ACNS 5.2.1 software and later releases, the Content Engine SSL terminates HTTPS requests in WCCP mode and in manual proxy mode if the requested HTTPS servers are configured on the Content Engine, and tunnels the rest of the HTTPS traffic.

About Tunneling of HTTPS Client Requests

Tunneling of HTTPS request is another mode that the ACNS software supports for HTTPS client requests. In this mode, the Content Engine can only support limited processing on the HTTPS client requests (for example, no caching support, and only limited filtering support).

The ACNS 3.0 software and later releases support HTTPS tunneling through the CONNECT method, which is the standard HTTPS tunneling method that is defined in the HTTP specification.

The ACNS 5.1.5 software and later releases support tunneling of transparently redirected native HTTPS traffic. This means that the Content Engine can accept native HTTPS traffic from the client and tunnel such requests to the origin HTTPS server. Even though the Content Engine can apply filtering on such traffic, it does not see the actual HTTPS content because the certificate and private key of the origin HTTPS server is not installed on the Content Engine.

Configuring HTTPS Proxy Caching for Standalone Content Engines

With HTTPS proxy caching, direct proxy routing is used to direct HTTPS requests (HTTPS-over-HTTP from client browsers) directly to the Content Engine. The Content Engine functions as a nontransparent forward proxy server (nontransparent HTTPS proxy server) for these client browsers. For information about how to configure a client browser to point directly to a Content Engine, see the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.

Configure the Content Engine to operate in HTTPS proxy mode to allow the Content Engine to service HTTPS requests from client browsers that have been configured to use this Content Engine as their HTTPS proxy server.
Note

In order to support incoming HTTPS proxy requests, a DNS server must be configured (as described in the “Configuring DNS Servers for the DNS Caching Service (Service 53)” section on page 7-64).

From the Content Engine GUI, choose Caching > HTTPS Proxy, and use the displayed HTTPS Proxy window. To obtain more information about how to use the HTTPS Proxy window to configure HTTPS proxy caching, click the HELP button in the window.

From the Content Engine CLI, use the https proxy global configuration command, as described in Table 7-8. The order in which the CLI commands are entered is not important.

### Table 7-8 HTTPS Proxy Features and the Related CLI Command

<table>
<thead>
<tr>
<th>Content Engine CLI Commands (Abbreviated Syntax)</th>
<th>HTTPS Proxy Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>https proxy incoming ports 1–65535, ports, . . .</td>
<td>Supports up to eight incoming proxy ports.</td>
</tr>
<tr>
<td>https proxy incoming ports 1–65535 wccp custom-web-cache . . .</td>
<td>Configures HTTPS incoming proxy ports and the custom-web-cache service to intercept HTTPS-over-HTTP requests.</td>
</tr>
<tr>
<td>no https destination-port allow 443 563 or https destination-port deny all</td>
<td>Denies unwanted access to any destination HTTPS port.</td>
</tr>
<tr>
<td>proxy-protocols outgoing-proxy exclude list word https proxy outgoing host {hostname</td>
<td>ip_address} port 1–65535</td>
</tr>
<tr>
<td>proxy-protocols transparent default-server</td>
<td>Uses the default outgoing HTTPS proxy server, if available.</td>
</tr>
<tr>
<td>proxy-protocols transparent original-proxy</td>
<td>Uses the outgoing HTTPS proxy server from the original request.</td>
</tr>
<tr>
<td>proxy-protocols transparent reset</td>
<td>Returns the incoming HTTPS request to the sending client during a cache miss.</td>
</tr>
</tbody>
</table>

The following example shows how to configure a standalone Content Engine (Content Engine A) to support HTTPS-over-HTTP proxy caching:

**Step 1** Configure the port numbers for the incoming proxy-mode requests.

    ContentEngineA(config)# https proxy incoming ports

where *ports* are the ports (in addition to port 80) on which the standalone Content Engine will accept incoming HTTPS requests from client browsers. This number ranges from 1 to 65535. You can specify up to eight ports.

This example configures the Content Engine to accept HTTPS requests on ports 8080, 8081, and 9090: If multiple ports are entered, separate each port number with a blank space.

    ContentEngineA(config)# https proxy incoming 8080 8081 9090

Note If multiple ports are entered, separate each port number with a blank space.
Remember that if you use this command to configure the Content Engine to accept HTTPS-over-HTTP requests on a port other than port 80, you must also configure the client browsers to send their HTTPS requests to that port on the Content Engine. For more information, see the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.

**Step 2** Designate a proxy server as the primary outgoing HTTPS proxy server for the Content Engine:

```bash
ContentEngineA(config)# https proxy outgoing host host port primary
```

- `host` is the hostname or IP address of the parent cache (outgoing HTTPS proxy server) to which HTTPS missed traffic is directed.
- `port` is the port number used by the parent cache to accept cache miss HTTPS-over-HTTP requests from the Content Engine.

Use the `primary` keyword to set the specified host as the primary outgoing HTTPS proxy server. If several servers (hosts) are configured with the `primary` keyword, the last one configured becomes the primary outgoing HTTPS proxy server for the Content Engine.

In this example, Content Engine A is configured to send its cache miss HTTPS traffic (cache misses for browser HTTPS-over-HTTP requests) to the host 10.1.1.1 on port 8088. Host 10.1.1.1 is explicitly designated as the primary outgoing HTTPS proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing HTTPS proxy server for Content Engine A.

```bash
ContentEngineA(config)# https proxy outgoing host 10.1.1.1 8088 primary
ContentEngineA(config)# https proxy outgoing host 10.1.1.2 220
```

In the ACNS 5.1.x software and earlier releases, you could only configure the Content Engine to use one outgoing HTTPS proxy server. With the ACNS 5.2.1 software and later releases, you can specify up to eight outgoing proxy servers for HTTPS-over-HTTP proxy requests. The benefit of supporting up to eight outgoing HTTPS proxy servers is that if one outgoing proxy fails, the Content Engine will fail over to the next specified outgoing proxy server, thereby providing redundancy. All outgoing requests will be directed to the primary outgoing proxy server. If the primary proxy server fails, requests are directed to the next active proxy server on the list.

**Step 3** View the current state of each HTTPS proxy server that is currently configured on the standalone Content Engine.

```bash
ContentEngineA# show https proxy
```

---

**Configuring HTTPS Transparent Caching for Standalone Content Engines**

To configure a Content Engine to support HTTPS transparent caching, you must configure the Content Engine and a router support the WCCP Version 2 HTTPS caching service (the `https-cache` service [service 70]). The `https-cache` service is the WCCP HTTPS caching service that permits WCCP Version 2-enabled routers to intercept port 443 TCP traffic and redirect this HTTPS traffic to the Content Engine (that is acting as a transparent forward proxy server, which is configured for HTTPS transparent caching). The Content Engine retrieves the requested content, stores a copy locally (performs HTTPS transparent caching) if the content is cacheable, and serves the requested content to the client.
In the ACNS 5.1.x software, only one interception mode (the accept-only mode) was supported for the https-cache service. With the accept-only mode, the Content Engine can only accept requests that are directed to configured HTTPS servers.

```
ContentEngine(config)# wccp https-cache router-list-num 1
```
or
```
ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list 1 https-cache
```

In both of the preceding examples, the Content Engine would only accept redirected HTTPS traffic if the HTTPS server was configured on the Content Engine (that is, you have used the https server global configuration command to specify the IP address or hostname and the private key and certificate of the origin HTTPS server on the Content Engine).

In the ACNS 5.2 software, another interception mode (the accept-all mode) was added for the https-cache service (service 70). When the accept-all mode is enabled, the Content Engine intercepts all HTTPS requests regardless of whether the origin HTTPS server is configured on the Content Engine. If the private key or certificate of the origin HTTPS server is not configured on the Content Engine, the Content Engine tunnels the request to the origin HTTPS server instead of SSL terminating the request. (In the ACNS 5.1 software, the Content Engine would bypass HTTPS requests that were directed to HTTPS servers that it had not been explicitly configured to support.)

The accept-all mode was added to support the filtering of HTTPS traffic.

```
ContentEngine(config)# wccp https-cache ?
accept-all       Accept all https traffic by default
mask             Specify mask used for CE assignment
router-list-num  Router list number
```

The accept-all mode works the same way as the traditional WCCP services do (for example, the standard web-cache service [service 0] that intercepts all web traffic by default). If the `wccp https-cache accept-all` option is used, the HTTPS cache (the Content Engine that has the https-cache service configured and enabled) operates in accept all mode (that is, all HTTPS traffic is intercepted by the Content Engine). If the `wccp https-cache accept-all` option is not used, the Content Engine operates in accept only mode.

The Content Engine listens for redirected HTTPS requests on the standard HTTPS port (default port 443). To intercept HTTPS traffic on ports other than the default port, configure a user-defined WCCP service (services 90 to 97). For more information on this topic, see the “Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 6-15.

The following example shows how to configure HTTPS transparent caching on a standalone Content Engine and a single router running WCCP Version 2:

**Step 1**
Configure the router to support the https-cache service (service 70).

a. Enable WCCP Version 2 on the router (for example, Router A).

```
RouterA# configure terminal
RouterA(config)# ip wccp version 2
```

b. Configure Router A to run the https-cache service (service 70).

```
RouterA(config)# ip wccp 70
```

c. Configure Router A to intercept all outgoing HTTPS traffic.

```
RouterA(config)# ip wccp 70 redirect out
```
Step 2  Configure the Content Engine to support the https-cache service.

a. Configure the list of routers that will be used to support the https-cache service. In this case, router list number 1 is created and consists of only one router (Router A, which has an IP address of 10.2.202.1).

   ContentEngine(config)# wccp router-list 1 10.1.202.1

b. Configure the Content Engine to accept transparently redirected HTTPS requests from the routers listed in router list 1 (Router A).

   ContentEngine(config)# wccp https-cache router-list-num 1

c. Enable WCCP Version 2 on the Content Engine.

   ContentEngine(config)# wccp version 2

d. If you want the Content Engine to intercept all HTTPS traffic and tunnel the HTTPS traffic for which it does not have a private key or certificate, then enable the accept-all mode on the Content Engine. This feature is typically used for filtering purposes (for example, to enable the Content Engine to use SmartFilter or Websense software to filter tunneled HTTPS requests).

   ContentEngine(config)# wccp https-cache accept-all

Step 3  Load the private key or certificates of the HTTPS origin servers from which the Content Engine will cache content (that is, act as the HTTPS origin server) instead of tunneling the HTTPS requests to the origin HTTPS server.

To load the private key or certificate on to the Content Engine, use the https EXEC command.

   ContentEngine# https ?
   cert       Certificate management commands
   certgroup  Certificate chain management commands
   key        Private key management commands

For more information, see the “Configuring Certificates and Private Keys for HTTPS Caching” section on page 7-32.

Step 4  (Optional). Create a certificate group on the Content Engine. This feature is used for HTTPS requests that are SSL terminated by the Content Engine. For more information, see the “Configuring Certificate Groups for HTTPS Caching” section on page 7-33.

Step 5  Configure the basic HTTPS server settings on the Content Engine. These basic settings must be set in order to enable HTTPS transparent caching on a Content Engine.

a. Use the https server server-name global configuration command to specify the server name of an origin HTTPS server from which the Content Engine caches content. For example, in the following example, the origin HTTPS server named “abc1” is configured on the Content Engine. After you specify the server name, the submode for HTTPS configuration is invoked and the prompt changes to ContentEngine(config-https)#.

   ContentEngine(config)# https server abc1
   ContentEngine(config-https)#
b. From HTTPS configuration submode, enter the certificate, the private key, and hostname of the HTTPS server (abc1), and then enable these settings (enter **enable** from the submode, as shown below) on the Content Engine. These are the minimal settings that you need to specify to enable caching of content of the specified HTTPS server.

```
ContentEngine(config-https)# cert ?
  WORD Certificate name
ContentEngine(config-https)# key ?
  WORD Private key name
ContentEngine(config-https)# host ?
  WORD FQDN or ip address of the origin HTTPS server
ContentEngine(config-https)# enable
```

Use the **cert** and **key** command options to configure a Content Engine to use a set of SSL certificates and keys that will enable the Content Engine to act as the origin HTTPS server. The Content Engine will be able to decode HTTPS traffic from a client and perform normal HTTP operations on it, such as caching and request processing. The Content Engine will be able to initiate HTTPS connections to an origin server and fetch content from origin servers upon cache miss (or cache validation). For more information, see the “Configuring HTTPS Server Settings on Standalone Content Engines” section on page 7-31.

**Step 6** (Optional) Configure certain advanced HTTPS server settings on the Content Engine to provide more detailed control of various aspects of SSL communication.

a. Configure certificate groups. For more information, see the “Configuring Certificate Groups for HTTPS Caching” section on page 7-33.

b. Configure the Content Engine to use SSL Version 2 only. Enter the **protocol-version sslv2-only** command option or the **https server name protocol-version sslv2-only** global configuration command:

```
ContentEngine(config-https)# protocol-version ?
  sslv2-only Only use and understand SSL v2 protocol
  sslv23-tlsv1 Use and understand SSL v2/v3 and TLS v1 protocols (default)
  sslv3-only Only use and understand SSL v3 protocol
  tlsv1-only Only use and understand TLS v1 protocol
```

or

```
ContentEngine(config)# https server name protocol-version ?
  sslv2-only Only use and understand SSL v2 protocol
  sslv23-tlsv1 Use and understand SSL v2/v3 and TLS v1 protocols (default)
  sslv3-only Only use and understand SSL v3 protocol
  tlsv1-only Only use and understand TLS v1 protocol
```

c. (Optional) Change the default HTTPS server authentication settings to meet the needs of your particular environment. For example, use the **serverauth ignore** command option (shown below), or the **https server name serverauth ignore** global configuration command to ignore particular errors in HTTPS server authentication.

```
ContentEngine(config-https)# serverauth ignore ?
  cert-not-yet-valid Ignore errors caused by using the certificate be valid
  domain-name Ignore errors due to domain name mismatch
  expired-date Ignore certificate expiration errors
  invalid-ca Ignore errors caused by an unrecognized CA
```
d. (Optional) Change the default session cache settings, and specify settings that meet your specific requirements by using the `session-cache` command option or the `https server name session-cache` global configuration command. For example, use the `size` option to specify the SSL session cache size setting. The default is 10,000 entries. Valid values are 0 to 20,000 entries.

```
ContentEngine(config-https)# session-cache ?
  size  SSL session cache size setting
  timeout  SSL session cache timeout setting
```

---

**Configuring HTTPS Server Settings on Standalone Content Engines**

After you have specified the name of an origin HTTPS server on a Content Engine using the `https server server-name` global configuration command, you can specify a set of parameters for the configured HTTPS server (see the following example). You can specify these parameters from the HTTPS configuration submode or from global configuration mode.

```
ContentEngine(config)# https server abcl
ContentEngine(config-https)# ?
cert              Select certificate to use for the HTTPS server
certgroup         Select certificate chains for the HTTPS server
enable            Enable caching of the HTTPS server
host              Input hostname or ip address of the origin HTTPS serve
key               Select private key to use for the HTTPS server
protocol-version  SSL protocol versions for client/server communication
serverauth        HTTPS server authentication commands
session-cache     SSL session caching parameters tuning
```

Table 7-9 describes the global configuration commands for configuring the HTTPS server settings on a standalone Content Engine.

**Table 7-9 CLI Commands for Configuring HTTPS Transparent Caching**

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>https server name cert</code></td>
<td>Configures a Content Engine to use a set of SSL certificates and keys to let the Content Engine act as the origin HTTPS server. The Content Engine can decode HTTPS traffic from a client and perform normal HTTP operations on it, such as caching and request processing. The Content Engine can initiate HTTPS connections to an origin server and fetch content from origin servers upon cache miss (or cache validation).</td>
</tr>
<tr>
<td><code>https server name key</code></td>
<td>Specifies the IP address for the origin HTTPS server. Use the IP address of the Content Engine in a central office acting as the HTTPS server when using this command. The <code>https server name enable</code> global configuration command enables the use of this HTTPS server.</td>
</tr>
<tr>
<td><code>https server name host</code></td>
<td>Sets the SSL protocol version used to control communication between the client and the HTTPS server.</td>
</tr>
<tr>
<td><code>https server name protocol-version</code></td>
<td></td>
</tr>
</tbody>
</table>
Table 7-9 CLI Commands for Configuring HTTPS Transparent Caching (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>https server name</td>
<td>Allows the use of authentication to reach the HTTPS server. You can also configure the authentication to ignore authentication errors such as invalid certification, domain name mismatches, certificate expiration errors, and unrecognized Certificate Authorities (CAs). By default, HTTPS server authentication is enabled on the Content Engine. Use the ignore command option to ignore particular errors in HTTPS server authentication.</td>
</tr>
<tr>
<td>session-cache</td>
<td>Specifies the size and timeout for the SSL session cache on the Content Engine. Use the size option to specify the SSL session cache size setting. The default is 10,000 entries. Valid values are 0 to 20,000 entries. Use the timeout option to specify the SSL session cache timeout setting. The default is 3,600 seconds. Valid values are 1 to 86,400 seconds.</td>
</tr>
</tbody>
</table>

Table 7-10 Parameters for the https cert and https key EXEC Commands

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cert</td>
<td>Enables creating, removing, and importing certificates.</td>
</tr>
<tr>
<td>cert-name</td>
<td>Name of the certificate.</td>
</tr>
<tr>
<td>add-cert</td>
<td>Adds a certificate from an external source.</td>
</tr>
<tr>
<td>create</td>
<td>Defines the name for a certificate.</td>
</tr>
<tr>
<td>remove</td>
<td>Removes a certificate with a given name.</td>
</tr>
<tr>
<td>key</td>
<td>Enables creating, removing, and importing a private key.</td>
</tr>
<tr>
<td>key_name</td>
<td>Name of the public and private key pair.</td>
</tr>
<tr>
<td>create</td>
<td>Defines the name for a private key.</td>
</tr>
<tr>
<td>import</td>
<td>Imports a private key from an external source.</td>
</tr>
<tr>
<td>remove</td>
<td>Removes a key with a given name.</td>
</tr>
<tr>
<td>url</td>
<td>Enables the use of a URL to point to the location of the private key or certificate.</td>
</tr>
<tr>
<td>URL</td>
<td>URL (HTTP, FTP-over-HTTP, or HTTPS) that points to the location of the private key or certificate.</td>
</tr>
</tbody>
</table>

Configuring Certificates and Private Keys for HTTPS Caching

In the ACNS 5.1. software and later releases, the Content Engine CLI can be used to configure certificates and private keys on the Content Engine in order to support HTTPS caching. (The Content Engine GUI does not currently support this configuration capability.)

Use the https EXEC command to create, remove, and import certificates and private keys when using a standalone Content Engine as an HTTPS server. Table 7-10 describes the parameters for the https cert cert-name and https key key_name EXEC commands.
You can assign a certificate and associate a key with the HTTPS server assuming that you have configured the Content Engine with the `https server` global configuration command. The Content Engine presents the certificate to HTTPS clients that make requests to the HTTPS server.

Use the `https cert` EXEC command to create certificates with a given name, or to import a certificate from external sources, or to remove existing certificates from the Content Engine.

Note: Private keys and certificates must be in the Privacy-Enhanced Mail (PEM) format. If the private keys and certificates are in Public Key Cryptography Standards 12 (PKCS12) format, the Content Engine will convert them internally to PEM format when importing the private keys or certificates.

---

## Configuring Certificate Groups for HTTPS Caching

In the ACNS 5.1 software and later releases, the Content Engine CLI can be used to configure certificate groups on the Content Engine in order to support HTTPS caching. (The Content Engine GUI does not currently support this configuration capability.)

Certificate groups are formed to represent a trust relationship chain from root Certificate Authority to end entity. Each one of the certificates in a certificate group except the end entity’s certificate signs and trusts the next one in the chain. An end entity’s certificate can be trusted only if all certificates in the certificate group leading to this certificate can be trusted. A certificate group can be used to represent an HTTPS server just like a single certificate, but with the added benefit that the client does not need to have all certificates locally. A certificate group can also be used to verify and authenticate an HTTPS server by comparing the server’s certificates to the certificate group.

Use the `https certgroup cert-name` EXEC command to create, remove, or form certificate groups on a standalone Content Engine. Table 7-11 describes the parameters for the `https certgroup cert-name` EXEC command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>certgroup</code></td>
<td>Enables adding, creating, or removing a certificate group.</td>
</tr>
<tr>
<td><code>cert-name</code></td>
<td>Name of the certificate group.</td>
</tr>
<tr>
<td><code>import</code></td>
<td>Adds a chain of certificates to the certificate group.</td>
</tr>
<tr>
<td><code>create</code></td>
<td>Creates a certificate group with the specified name.</td>
</tr>
<tr>
<td><code>remove</code></td>
<td>Removes a certificate group with the specified name.</td>
</tr>
</tbody>
</table>
Chapter 7
Configuring Conventional Caching Services for Standalone Content Engines

Configuring HTTPS Outgoing Proxy Servers for Standalone Content Engines

When you use the `https proxy outgoing host` global configuration command to configure the Content Engine to use an HTTPS outgoing proxy, all incoming HTTPS requests are directed to this outgoing proxy. The `proxy-protocols outgoing-proxy exclude` global configuration 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 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 not HTTPS, 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` global configuration command is invoked, the Content Engine addresses the request to the destination server directly and not to the client’s intended proxy server. However, if the `proxy-protocols transparent reset` global configuration command is used on the Content Engine and a cache miss occurs, all transparently intercepted requests sent by clients are returned to the client and requested objects are not delivered. For more information about configuring HTTPS outgoing proxy exclusion settings on standalone Content Engines, see the “Configuring HTTP and HTTPS Outgoing Proxy Exclusion Settings” section on page 14-5.

The following example shows how to configure the Content Engine (which is acting as a nontransparent forward proxy) as an HTTPS proxy server and have it listen on port 8081 for HTTPS requests from client browsers:

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

In this example, Content Engine A is configured to send its missed HTTPS traffic (cache misses for browser requests for HTTPS content [HTTPS-over-HTTP requests]) to the host 10.1.1.1 on port 8088. Host 10.1.1.1 is explicitly designated as the primary outgoing HTTPS proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing HTTPS proxy server for Content Engine A.

```
ContentEngineA(config)# https proxy outgoing host 10.1.1.1 8088 primary
ContentEngineA(config)# https proxy outgoing host 10.1.1.2 220
```

In the ACNS 5.1.x software and earlier releases, you could only configure the Content Engine to use one outgoing HTTPS proxy server. In the ACNS 5.2.1 software release, the ability to configure the Content Engine to use up to eight outgoing HTTPS proxy servers was added. For more information on this topic, see Chapter 14, “Configuring Primary and Backup Proxy Servers for Standalone Content Engines.”

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 as defined by the `proxy-protocols transparent` global configuration command. For more information on this topic, see the “Configuring HTTP and HTTPS Outgoing Proxy Exclusion Settings” section on page 14-5.
Preventing Unwanted Access to Any Destination Port

A Content Engine may not provide the desired level of security if rules controlling access (allow or deny) to destination ports are not configured on the Content Engine.

**Caution**

The Content Engine may not provide the desired level of security if the policies controlling access to the destination ports have not been configured. Consequently, we strongly recommend that you set restrictions that allow or deny HTTPS traffic to destination ports. Default settings may not provide the desired level of security.

In order to prevent web users from creating HTTPS tunnels to non-HTTPS servers, access to destination ports can be restricted. The restrictions can be made for individual ports or for all ports. In the case of a conflict in the restriction rules, the individual port setting takes precedence over the “all” rule, and the “allow” rule takes precedence over the “deny” rule.

Preventing access to destination ports is supported with WCCP transparent redirection (that is, a WCCP Version 2-enabled router transparently intercepts and redirects HTTPS request to the Content Engine) as well as with direct proxy routing, in which client browsers send their HTTPS requests directly to the Content Engine.

You can use the Content Engine GUI or CLI to configure this feature. To use the Content Engine GUI, choose **Cache > HTTPS Proxy**, and use the displayed HTTPS Proxy window to restrict destination ports. To obtain more information about how to use the HTTPS Proxy window, click the **HELP** button in the window.

To use the Content Engine CLI, enter this command:

```
ContentEngine(config)# https destination-port ?
allow  Allow HTTPS traffic to port (443 and 563 allowed by default)
deny  Deny HTTPS traffic to port (port under 1024 denied by default)
```

To prevent unwanted access to any destination HTTPS port when a request is going through the Content Engine, use the following command sequence:

```
ContentEngine(config)# no https destination-port allow 443 563
ContentEngine(config)# https destination-port deny all
```

This command sequence denies access to any port above and below 1024. Ports 443 and 563 (the standard HTTPS ports) must be explicitly denied access using the **no https destination-port allow 443 563** global configuration command.

**Note**

TCP and UDP packets use port numbers defined by the application in use. Typically, the port range 0–255 is used for standard public applications such as FTP, and the port range 256–1023 is used by companies for nonstandard applications. For instance, FTP uses port 21, and Telnet uses port 23. Port numbers from 1024 through 65,536 are unregulated, so it is best to specifically deny access through any port number.

For example, when these commands are configured on the Content Engine and the request to access port xxx at https://banking.bankabc.com is redirected to this Content Engine through WCCP transparent redirection or direct proxy routing, the connection to port xxx is denied.
Configuring FTP Caching for Standalone Content Engines

This section provides an overview of the FTP caching feature for standalone Content Engines and describes how to configure the different types of FTP caching:

- **Overview of FTP Caching with Standalone Content Engines**, page 7-36
- **Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines**, page 7-38
- **Configuring FTP Native Caching for Standalone Content Engines**, page 7-41

Note: You can use the Content Engine GUI or the CLI to configure FTP caching on a standalone Content Engine.

Overview of FTP Caching with Standalone Content Engines

A standalone Content Engine can be configured for FTP caching in the following two usage modes:

- **FTP-over-HTTP mode**—The standalone Content Engine (acting as a nontransparent forward proxy server) caches the contents of the specified FTP URLs that are sent to it directly by clients who are using the HTTP protocol. This allows users to use their browsers running the HTTP protocol to send and receive files on remote FTP servers. For more information, see the “Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines” section on page 7-38.

- **Native FTP mode**—The standalone Content Engine caches the contents of the FTP request that are sent from clients in the native FTP protocol. In the ACNS 5.3.1 software and later releases, native FTP caching is supported in transparent and nontransparent proxy mode. (Native FTP caching was only supported in transparent proxy mode in the ACNS 5.1 and 5.2 software releases.) For more information, see the “Configuring FTP Native Caching for Standalone Content Engines” section on page 7-41.

In both of these usage modes, the Content Engine uses the FTP protocol to retrieve and locally cache the content of the FTP requests. These two usage modes differ in the protocol used by the client to issue the FTP request. In FTP-over-HTTP mode, clients use their browsers (the HTTP protocol) to issue FTP requests. In native FTP mode, clients use the native FTP protocol to issue FTP requests, as shown in the following example:

```
ContentEngine# ftp server.cisco.com
```
Table 7-12 summarizes the usage modes and the types of supported FTP caching.

**Table 7-12  Usage Modes and Types of Supported FTP Caching**

<table>
<thead>
<tr>
<th>Usage Mode</th>
<th>Description</th>
<th>Transparent Redirection</th>
<th>Direct Proxy Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP-over-HTTP</td>
<td>Content Engine caches the contents of the specified FTP URLs that are sent to it directly by clients who are using the HTTP protocol.</td>
<td>Not supported</td>
<td>Client browsers send their nontransparent FTP-over-HTTP requests directly to the Content Engine (a nontransparent forward proxy server). This type of caching is called nontransparent FTP-over-HTTP caching, and is supported in the ACNS 5.0 software and later releases. For more information, see the “Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines” section on page 7-38.</td>
</tr>
<tr>
<td>Native FTP</td>
<td>Content Engine caches the contents of the FTP requests that are sent from FTP clients in the native FTP protocol</td>
<td>WCCP Version 2-enabled routers transparently intercept FTP native requests from FTP clients, and redirect these requests to Content Engines (transparent proxy servers). This type of caching is called transparent FTP native caching, and is supported in the ACNS 5.1 software and later releases. For more information, see the “Configuring Transparent FTP Native Caching” section on page 7-54.</td>
<td>FTP clients send their nontransparent FTP native requests directly to the Content Engine (a nontransparent proxy server). This type of caching is called nontransparent FTP native caching, and is supported in the ACNS 5.3.1 software and later releases. For more information, see the “Configuring Nontransparent FTP Native Caching” section on page 7-42.</td>
</tr>
</tbody>
</table>

**Note**

Transparent redirection of FTP requests is supported only by WCCP Version 2; transparent redirection through a Layer 4 switch is not supported.
In the ACNS 5.4.1 software release, the authentication option was added to the ftp-native proxy configuration command to support proxy authentication for nontransparent FTP native requests. For more information about this topic, see the “Configuring Request Authentication for Nontransparent FTP Native Requests” section on page 10-55.

In the ACNS 5.4.1 software release, the access-list option was added to the ftp-native configuration command to support IP ACLs to grant or deny access to the native FTP proxy service that is running on the Content Engine:

ContentEngine(config)# ftp-native ?
access-list Configure access-lists for ftp-native proxy-mode and transparently redirected connections
object Configuration of FTP object
proxy Configuration for proxy-mode requests

For more information about this topic, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.

In the ACNS 5.3.1 software release, the ftp keyword was replaced with the ftp-over-http and ftp-native keywords to clearly differentiate between FTP native caching and FTP-over-HTTP caching:

ContentEngine(config)# ftp-over-http ?
age-multiplier FTP-over-HTTP caching heuristic modifiers
max-ttl Maximum time to live for objects in the cache
min-ttl Minimum time to live for objects in the cache (in minutes, default is 30)
object Configuration of FTP object
proxy Configuration for incoming proxy-mode requests
reval-each-request Configuration of revalidation for every request

In the ACNS 5.3.1 software release, the show ftp proxy EXEC command was replaced with the show ftp-over-http and show ftp-native EXEC commands. In the ACNS 5.3.1 software release, the show statistics ftp EXEC command was replaced with the show statistics ftp-over-http and show statistics ftp-native EXEC commands. In the ACNS 5.3.1 software release, the clear statistics ftp EXEC command was replaced with the clear statistics ftp-over-http and clear statistics ftp-native EXEC commands.

### Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines

In the ACNS 5.0 software release, support for the proxying and caching of FTP-style requests over HTTP in proxy mode was added. When the Content Engine is configured in proxy mode, it can handle FTP-style requests over HTTP transport. When the Content Engine receives an FTP request from a client, it processes the request by searching its cache. If the object is not in its cache, it retrieves the object from an upstream FTP proxy server if this proxy server has been configured, or it retrieves the object directly from the origin FTP server.

With nontransparent FTP-over-HTTP caching, the standalone Content Engine is functioning as a nontransparent forward proxy server for FTP-over-HTTP requests from client browsers. The ACNS 5.1 software and later releases support proxying and caching of FTP URL client requests using proxy-mode HTTP requests when URLs specify the FTP protocol (for example, ftp://ftp.mycompany.com/ftpdir/ftp_file).

The following example of an FTP-over-HTTP request shows how the end user can use a browser to access public files from an FTP server:

For these requests, the client uses HTTP as the transport protocol with the Content Engine, and the Content Engine uses FTP with the FTP server. 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 retrieves the object from an upstream FTP proxy server (if one is configured in the ACNS 5.2.1 software and later releases) 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 FTP proxy supports commonly used MIME types, attaches the corresponding header to the client, chooses the appropriate transfer type (binary or ASCII), and enables the browser to open the FTP file with the configured application. For unknown file types, the proxy uses binary transfer as the default and instructs the browser to save the download file instead of opening it. The FTP proxy returns a formatted directory listing to the client if the FTP server replies with a known format directory listing. The formatted directory listing has full information about the file or directory and provides the ability for users to choose the download transfer type.

To use the Content Engine CLI to configure nontransparent FTP-over-HTTP caching for standalone Content Engines, follow these steps:

**Step 1** Configure the port numbers for the incoming proxy-mode FTP-over-HTTP requests.

```
ContentEngine(config)# ftp-over-http proxy incoming ports
```

*ports* are the port numbers on which the standalone Content Engine will accept incoming FTP-over-HTTP requests from client browsers in addition to port 80. Valid port numbers are 1 to 65535. You can specify up to eight ports.

The following example configures the Content Engine to accept FTP-over-HTTP requests from client browsers on ports 8080, 8081, and 9090. Up to eight incoming proxy ports can be configured on the same command line:

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

If you enter an illegal port number, a message appears informing you of this situation. For example, if you attempt to specify port 9002 as the incoming port for FTP-over-HTTP requests, you are informed that this port is reserved for the RealProxy application, as follows:

```
ContentEngine(config)# ftp-over-http proxy incoming 9002
Port 9002 is reserved for application the Real_Proxy.
```

Remember that if you use the *ftp-over-http proxy incoming* command to configure the Content Engine to accept FTP-over-HTTP requests on a port other than port 80, you must also configure the client browsers to send their FTP-over-HTTP requests to that port. For more information, see the “Pointing Client Browsers Directly to a Standalone Content Engine” section on page 4-36.

**Step 2** Configure transaction logging on the Content Engine.

The following example shows how to configure the Content Engine to use the Apache transaction log format then enable transaction logging on the Content Engine:

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

For more information on this topic, see the “Enabling Transaction Logging” section on page 21-33.
Chapter 7  Configuring Conventional Caching Services for Standalone Content Engines

Step 3  Configure FTP cache object freshness settings for FTP-over-HTTP caching. These parameters can be configured for either directory listings or particular objects in the cache.

Tip  The ACNS 5.x software can be tuned to balance HTTP and FTP object freshness with the cache hit rate. The ACNS software default parameters are weighted in favor of securing fresh content over maximizing the cache hit rate (to avoid the undesirable situation of increasing the cache hit rate by serving stale content). Text objects refer to HTML pages. Binary objects refer to all other web objects, such as GIFs and JPEGs.

a. Specify the maximum size of an FTP object that should be stored in the Content Engine cache for FTP-over-HTTP caching. In following example, the maximum size for an FTP object size is set to 2 megabytes for FTP-over-HTTP caching.

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

b. For FTP-over-HTTP caching, you can also use the `ftp-over-http age-multiplier`, `ftp-over-http max-ttl`, `ftp-over-http reval-each-request`, and the `ftp-over-http min-ttl` global configuration commands.

The following example configures a maximum Time To Live (TTL) of 3 days in the cache for directory listing objects and file objects for FTP-over-HTTP caching:

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

The following example configures the Content Engine to keep FTP objects in the cache for a minimum of 10 minutes and a maximum of 24 hours (1 day) for FTP-over-HTTP caching:

```
ContentEngine(config)# ftp-over-http min-ttl 10
ContentEngine(config)# ftp-over-http max-ttl hours directory-listing 24 file 24
```

c. Force the Content Engine to revalidate all objects for every FTP-over-HTTP request.

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

Note  In the ACNS 5.3.1 software release, the `ftp` keyword was replaced with the `ftp-over-http` and `ftp-native` keywords.

Step 4  Configure one or more outgoing FTP proxy servers for the Content Engine with the `ftp-over-http proxy outgoing host` global configuration command. Enter the hostname or IP address for the outgoing FTP proxy servers. The primary outgoing FTP proxy server is the parent cache (upstream FTP proxy server) to which you want this Content Engine to direct all of its missed FTP traffic without using ICP or WCCP. For more information, see the “Designating a Primary Outgoing FTP Proxy Server” section on page 14-3.

Step 5  Specify the password that has to be used during anonymous FTP-over-HTTP operation with the `ftp-over-http proxy anonymous-pswd` global configuration command.

Step 6  Enable active mode on this Content Engine for FTP-over-HTTP mode.

```
ContentEngine(config)# ftp-over-http proxy active-mode enable
```

In FTP-over-HTTP caching mode, if the `ftp-over-http proxy active-mode` global configuration command is used the Content Engine first attempts to use active mode with the origin FTP server for the data connection. If the active mode fails, the Content Engine attempts to use passive mode for the data connection.
In FTP-over-HTTP mode, if the `ftp-over-http proxy active-mode` command is not used the Content Engine first attempts to use passive mode with the FTP server for the data connection, and automatically switches to active mode if passive mode is not supported by the FTP server.

**Step 7** View the current FTP-over-HTTP configuration on the standalone Content Engine.

```
ContentEngine(config)# exit
ContentEngine# show ftp-over-http
```

**Step 8** View the statistics for the FTP-over-HTTP requests that this standalone Content Engine has handled.

```
ContentEngine# show statistics ftp-over-http
```

For example, the command output shows the number of FTP-over-HTTP requests received by the Content Engine, the number of FTP-over-HTTP hits and misses, as well as the number of FTP-over-HTTP requests that the Content Engine has forwarded to the origin FTP server or to the specified outgoing proxy server. The command output also shows the number of FTP-over-HTTP errors.

**Step 9** (Optional). Clear the FTP-over-HTTP statistics on the Content Engine.

```
ContentEngine# clear statistics ftp-over-http
```

---

**Configuring FTP Native Caching for Standalone Content Engines**

This section describes how to configure FTP native caching for standalone Content Engines:

- [Configuring Nontransparent FTP Native Caching, page 7-42](#)
- [Configuring Transparent FTP Native Caching, page 7-54](#)

For the ACNS 5.1 and 5.2 software releases, all of the following restrictions apply to FTP native caching support:

1. Maximum FTP object size of 200 megabytes
2. No support for bandwidth control for FTP client requests and FTP server pulls
3. No support for the Type of Service (ToS) bit for FTP client requests
4. No support for pre-positioned files in the cdnfs
5. No support for ICAP
6. No support for nontransparent proxy
7. No support for proxy authentication
8. No support for ICP
9. No support for healing mode
10. No support for Layer 4 switch FTP redirection
11. No support for FTP request proxy rules
12. No support for MIN-TTL and AGING-HEURISTIC-TTL cache control knob configurations
13. No support for any URL filtering schemes (good list, bad list, N2H2, Websense, and SmartFilter)
14. No support for caching files from a Macintosh FTP server
15. No support for offline operation of the FTP proxy server
For the ACNS 5.3.x software releases, all of the preceding 15 restrictions still apply except for restriction 6 (no support for nontransparent proxy). In the ACNS 5.3.1 software release, support of the nontransparent proxy was added for FTP native mode. For more information about this topic, see the “Configuring Nontransparent FTP Native Caching” section on page 7-42.

For the ACNS 5.4.x software releases, all of the preceding 15 restrictions still apply except for the following two restrictions:

- Restriction 6 (no support for nontransparent proxy)
- Restriction 7 (no support for proxy authentication for nontransparent native FTP requests)

In the ACNS 5.4.1 software release, support for proxy authentication (request authentication at the Content Engine) was added for nontransparent FTP native requests. In the ACNS 5.4.1 software release, the authentication option was added to the ftp-native proxy configuration command to support proxy authentication for nontransparent FTP native requests. For authentication of nontransparent FTP native requests, the ACNS 5.4.1 software and later releases support RADIUS, TACACS+, NTLM, and LDAP.

By default, the ftp-native proxy authentication feature is disabled. Because the FTP protocol is inherently insecure, the authentication credentials can be sniffed off the network and expose user credentials that otherwise would have been provided over a secure channel (for example, in the case of HTTP). You can enable the ftp-native proxy authentication feature as follows:

```
ContentEngine(config)# ftp-native proxy authentication enable
```

### About IP ACL Support for FTP Native Requests

In the ACNS 5.4.1 software and later releases, you can use IP ACLs to grant or deny access to the native FTP proxy service that is running on a standalone Content Engine. For more information about this topic, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.

### Configuring Nontransparent FTP Native Caching

In the ACNS 5.3.1 software and later releases, the ability to proxy and cache nontransparent FTP native requests is supported. With this capability, a Content Engine has the ability to handle client FTP native requests from FTP clients in proxy mode.

When the Content Engine receives an FTP native request from an FTP client (for example, an FTP native request from a Reflection X or WS-FTP client or a UNIX or DOS command line FTP program), the Content Engine processes the request. If the requested content is already stored in the local cache, the Content Engine serves the content to the FTP client. Otherwise, the Content Engine performs an FTP request to the origin FTP server to retrieve the requested content, and then stores the content in its local cache. This type of caching is called nontransparent FTP native caching. Native FTP requests are logged in the HTTP transaction log on the standalone Content Engine.

Both passive and active mode for retrieving files and directories is supported. In FTP native caching mode, if the ftp-native proxy active-mode enable global configuration command is used, the Content Engine uses the same mode with the FTP server for the data connection as the client used to access the Content Engine, which can be either active or passive. If the ftp-native proxy active-mode enable command is not specified, the Content Engine uses passive mode with the FTP server for the data connection.

**Note**

Passive mode transfer between the FTP client and the FTP native proxy cannot be done if the Content Engine is behind a NAT because the Content Engine will not know its conduit IP address.
In the ACNS 5.4.1 software and later releases, proxy authentication (that is, authentication at the Content Engine) for nontransparent FTP native requests is also supported. The username that is provided by the FTP client (for example, the Reflection X client) during the proxy authentication is logged in the transaction log if either the extended-squid or custom logging formats have been configured on the Content Engine.

To configure nontransparent FTP native caching on standalone Content Engines, follow these steps:

**Step 1** Configure the port numbers for the incoming proxy-mode FTP requests (FTP native requests).

```
ContentEngine(config)# ftp-native proxy incoming ports
```

*ports* are the port numbers on which the standalone Content Engine will accept incoming requests (native FTP requests) from FTP clients. Valid port numbers are 1 to 65535. You can specify up to eight incoming ports.

The following example configures the Content Engine to accept FTP native requests from FTP clients on eight ports (port 8501, 8502, 8503, 8504, 8505, 8506, 8507, and 8508). Up to eight incoming proxy ports can be configured on the same command line, as shown in the following example:

```
ContentEngine(config)# ftp-native proxy incoming 8501 8502 8503 8504 8505 8506 8507 8508
```

If you reenter the **ftp-native proxy incoming** command, the Content Engine is reconfigured and only uses the ports specified in the most recently specified **ftp-native proxy incoming** command. For example, if you enter the following **ftp-native proxy incoming** command, the Content Engine uses the eight specified ports as incoming proxy ports:

```
ContentEngine(config)# ftp-native proxy incoming 8501 8502 8503 8504 8505 8506 8507 8508
```

However, if you reenter the following **ftp-native proxy incoming** command, the Content Engine is reconfigured to use only port 8501 as an incoming proxy port and drops the other seven previously configured ports as incoming proxy ports:

```
ContentEngine(config)# ftp-native proxy incoming 8501
```

If you enter an illegal port number, a message appears informing you about this situation. For example, if you attempt to specify port 554 as the incoming port for proxy-mode FTP native requests, you are informed that this port is reserved for the RTSP gateway that runs on the Content Engine, as follows:

```
ContentEngine(config)# ftp-native proxy incoming 554
Port 554 is reserved for application the RTSP_Gateway.
```

**Step 2** Specify the maximum size of an FTP object that should be stored in the Content Engine cache for FTP native caching. This parameter can be configured for either directory listings or particular objects in the cache.

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

In the following example, the maximum size for an FTP object size is set to 2 MB for FTP native caching:

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

**Step 3** (Optional). Configure the Content Engine to use IP ACLs to permit or deny access to the native FTP proxy service that is running on the Content Engine:

```
ContentEngine(config)# ftp-native access-list in {std-acl-num | std-acl-name}
ContentEngine(config)# ftp-native access-list out {ext-acl-num | ext-acl-name}
```
For example, the following commands define a standard access list that grants the FTP clients on the 192.168.1.0 subnetwork access to the native FTP proxy service that is running on the Content Engine:

```
ContentEngine(config)# ip access-list standard 3
ContentEngine(config-standard-nacl)# permit 192.168.1.0 0.0.0.255
ContentEngine(config-standard-nacl)# exit
```

Associate this standard IP ACL (access list 3) with the native FTP proxy service, and activate this standard IP ACL on the Content Engine.

```
ContentEngine(config)# ftp-native access-list in 3
```

The Content Engine applies the specified IP ACL (for example, ACL_3) to native FTP inbound traffic.

**Note**

In the ACNS 5.4.1 software and later releases, you can use IP ACLs to grant or deny access to the native FTP proxy service that is running on a standalone Content Engine. For more information, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.

**Step 4** (Optional). Create and then download a custom acl-denied message to the Content Engine. The Content Engine will display this custom message to the FTP client (for example, Reflection X clients, WS-FTP clients, or clients of UNIX or DOS command line FTP programs) when it denies an incoming connection based on the IP ACLs that are defined for the native FTP proxy service.

a. Create the acl-denied message and save it as a text file (for example, save it as a file named warning.txt to the errors directory on the server named myserver.com). The file size of the custom acl-denied message should not exceed 16 KB.

The following example uploads the warning.txt file to the errors directory on the server named myserver.com:

```
ContentEngine# ftp-native custom-message upload acl-denied
http://www.myserver.com/errors/ftp-native-acl-denied.txt
```

b. Download the ftp-native-acl-denied.txt file to the Content Engine:

```
ContentEngine# ftp-native custom-message download acl-denied
http://www.myserver.com/errors/ftp-native-acl-denied.txt
```

c. Display the contents of the custom acl-denied message that has been downloaded to the Content Engine:

```
ContentEngine# show ftp-native custom-message acl-denied ftp-native-acl-denied.txt
```

**Step 5** (Optional). Configure the Content Engine to display a custom welcome message to welcome proxy mode connections from FTP clients.

a. Use a text editor to create a custom welcome message and save it as a text file (for example, welcome.com).

b. Use the HTTP, HTTPS, or FTP protocol to upload the welcome.txt file to a server. The file size of the custom welcome message should not exceed 16 KB.

The following example uploads the welcome.txt file to the mgmt directory on the server named myserver.com:

```
ContentEngine# ftp-native custom-message upload welcome
http://www.myserver.com/mgmt/welcome.txt
```
c. Download the welcome.txt file to the Content Engine:

```
ContentEngine# ftp-native custom-message download welcome
http://www.myserver.com/errors/welcome.txt
```

d. Verify that the downloaded file exists:

```
ContentEngine# show ftp-native custom-message
welcome
acl-denied
ContentEngine#
```

e. Display the contents of the downloaded custom welcome message:

```
ContentEngine# show ftp-native custom-message welcome
Welcome to the Content Engine that is acting as your FTP proxy.
Login to the proxy using the proxy username and password.
```

**Note** For more information, see the “Creating Custom Messages for FTP Proxy Responses for FTP Native Requests” section on page 5-19.

---

**Step 6** Use an FTP client (for example, a UNIX command line FTP program) to send a request to the Content Engine to verify that the custom welcome message is displayed when the Content Engine accepts an incoming FTP connection from an FTP client (for example, a Reflection X client, a WS-FTP client, or a UNIX or DOS command line FTP program).

In the following example, the FTP client sends the Content Engine (IP address 172.31.255.255) a native FTP request on port 8501. After accepting the incoming FTP connection, the Content Engine displays the custom welcome message to the FTP client:

```
shell# ftp -d 172.31.255.255 8501
Connected to 172.31.255.255
220 Welcome to the Content Engine that is acting as your FTP proxy. Login to the proxy using username and password.
```

**Step 7** Configure and enable transaction logging on the Content Engine.

In the ACNS 5.4.1 software and later releases, the username that is provided by the FTP client during the proxy authentication is logged in the transaction log if either the Extended Squid or Custom logging formats have been configured on the Content Engine:

The following example configures the Content Engine to use the Extended-Squid transaction log format, and then transaction logging is enabled on the Content Engine:

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

For the Custom transaction logging format, you must include the %u format-specifier when you configure the `transaction-logs format custom` command. For more information, see the “Enabling Transaction Logging” section on page 21-33.

**Step 8** Enable FTP native active mode on the Content Engine.

```
ContentEngine(config)# ftp-native proxy active-mode enable
```
Step 9  (Optional). Enable the FTP proxy authentication feature on the Content Engine.

By default, this feature is disabled. Because the FTP protocol is inherently insecure, the authentication credentials can be sniffed off the network and expose user credentials that otherwise would have been provided over a secure channel (for example, in the case of HTTP).

```
ContentEngine(config)# ftp-native proxy authentication enable
```

If you enter the `ftp-native proxy authentication enable` command and you have not already configured an authentication service (for example, RADIUS, LDAP, NTLM, or TACACS) on the Content Engine, a message is displayed. The message indicates that you must configure an authentication service on the Content Engine before you can enable the FTP proxy authentication feature.

Step 10  Configure the FTP clients (client side) to send their native FTP requests directly to the Content Engine. For more information, see the next section, “Configuring the Client Side of Nontransparent FTP Native Caching.”

Step 11  View the current FTP native proxy configuration.

```
ContentEngine# show ftp-native
```

Step 12  Display the statistics for the FTP native requests that this standalone Content Engine has handled.

```
ContentEngine# show statistics ftp-native
```

The command output shows the number of FTP native GET requests received by the Content Engine, the number of FTP native hits and misses for GET requests, as well as the number of FTP native PUT requests that have been received by this Content Engine. In the ACNS 5.4.1 software and later releases, the command output also shows the number of transparent FTP native requests, the number of nontransparent FTP native requests, the number of native FTP proxy authentication requests, and the number of failed native FTP proxy authentication requests.

Step 13  Clear the FTP native statistics on the Content Engine.

```
ContentEngine# clear statistics ftp-native
```

---

**Configuring the Client Side of Nontransparent FTP Native Caching**

Content Engines that are acting as nontransparent FTP proxy servers can accept FTP native requests from such FTP clients as Reflection X clients, WS-FTP clients, and UNIX or DOS command line FTP programs. This section provides some examples of how to configure these different types of FTP clients to send FTP native requests directly to the Content Engine, which is acting as a nontransparent FTP proxy server for these FTP clients.

**Note**

In the ACNS 5.4.1 software release, support for proxy authentication (that is, authentication at the Content Engine) was added for nontransparent FTP native requests. The username that is provided by the FTP client (for example, Reflection X clients, WS-FTP clients, or clients of UNIX or DOS command line FTP programs) during the proxy authentication process is logged in the transaction log if one of the following transaction logging formats have been configured on the Content Engine: Extended-Squid logging or Custom logging. For more information about the FTP proxy authentication feature, see the “Configuring Request Authentication for Nontransparent FTP Native Requests” section on page 10-55.
The following example shows how to configure the client side proxy FTP request to the Content Engine through the Windows-based Reflection X client software:

**Step 1**  From the Windows Start menu, choose **Programs > Reflection > FTP Client**. The Connect to FTP Site window appears. (See **Figure 7-2**.)

**Figure 7-2  Using Reflection X Client Software to Connect to an FTP Site**

**Step 2**  Select one of the FTP sites that are listed in the Connect to FTP Site window.

**Step 3**  Click the **Properties** button. The Properties window for the selected FTP site appears. (See **Figure 7-3**.)

**Figure 7-3  Properties Window**

**Step 4**  In the General tab, click the **Security** button. The Securities dialog box appears. (See **Figure 7-4**.)
Step 5 In the Securities dialog box, complete the following steps:
   a. Select the Proxy-server check-box (in the Proxy tab).
   b. Select the **Passthrough** entry from the Server type drop-down menu.

Step 6 Click the **Configure** button. The Passthrough Server dialog box appears. (See **Figure 7-5**.)

Step 7 In the Passthrough Server dialog box, complete the following steps:
   a. Select the **username@servername** entry in the Style drop-down menu.
   b. Enter the Content Engine’s hostname or IP address in the **Server name** edit-box.
   c. Ensure that the **Passthrough authentication** check box is not checked.
   d. Click **OK** to close the Passthrough Server dialog box, and to return to the Securities dialog box.

Step 8 In the Securities dialog box, click **OK** to return to close the Securities dialog box.

Step 9 In the Properties window (**Figure 7-3**), click the Connection tab.

Step 10 In the Connection tab (**Figure 7-6**), set the TCP port to the Content Engine’s FTP-native incoming proxy port.
In the TCP port field, you must enter the same port number that you specified with the `ftp-native proxy incoming port` global configuration command (for example, port 7780) in Step 1 of the “Configuring Nontransparent FTP Native Caching” section on page 7-42.

**Step 11** In the Connection tab, click **Apply** and then click **OK** to return to the Connect to FTP Site window. (See Figure 7-2.)

**Step 12** Verify that the Reflection X client can connect to the FTP site.

a. In the Connect to FTP Site window, click the **Connect** button. The Reflection FTP dialog box appears asking you to enter a password. (See Figure 7-7.)

**Figure 7-7    Reflection FTP Password Dialog Box**

b. Enter the password and click **OK**.
WS-FTP is a Windows-based FTP client for transferring files from a Windows-based desktop to remote machines. This licensed software from Ipswitch, Inc. allows you to perform single file transfers from a Windows-based computer. The following example shows how to configure the client side proxy FTP request to the Content Engine through the Windows-based WS-FTP client software:

**Step 1** From the Windows Start menu, choose Programs > Ipswitch WS_FTP_Home > WS_FTP Home. The WS-FTP Home window appears. (See Figure 7-8.)

**Figure 7-8 Using WS-FTP Client Software to Connect to an FTP Site (an FTP Server)**

**Step 2** From the File menu, select Connect > Connection Wizard.

**Step 3** Enter the site name (the name of the FTP server that you want to connect to).

**Step 4** Enter the IP address of the FTP server.

**Step 5** Enter the username and password of the FTP server.

**Step 6** Select FTP as the connection type.

**Step 7** Click the Finish button.

**Step 8** From the Tools menu, select Options > Firewall.

**Step 9** Click the New button.

**Step 10** Enter the name and IP address of the Content Engine (that is acting as this client’s nontransparent proxy server).

**Step 11** Select Script (fwsc) as the type. This option specifies that you want to use a firewall script.

**Step 12** Enter the proxy port number.
You must enter the same port number that you specified with the `ftp-native proxy incoming port` global configuration command (for example, port 7780) in Step 1 of the “Configuring Nontransparent FTP Native Caching” section on page 7-42.

**Step 13** Click **Okay**.

**Step 14** Click the Firescript editor to open the editor.

**Step 15** Create a firewall script.

**Step 16** In the Firescript editor, paste the content of the firewall script that you have created.

The following is an example of the content of a sample firewall script that you would paste into the Firescript editor:

```plaintext
/fwsc
author=cisco
version=1
verdate=2004.11.23
required=HostUserId,HostPassword,HostAddress
connectto=firewall

[comment]
This script is similar to the "site hostname" script except that the following line at the beginning of the script section is removed:
  send (" ") {}

[script]
send ("SITE %HostAddress") {}

send ("USER %HostUserId")
{
  case (300..399):
    continue;

  case any:
    return (false);
}

send ("PASS %HostPassword")
{
  case (200..299) and contains(lastreply, "ACCOUNT") and not isempty(HostAccount):
    continue;

  case (300..399):
    continue;

  case (200..299):
    jump success;

  case any:
    return (false);
}

send ("ACCT %HostAccount")
{
  case (200..299):
    jump success;

  case any:
    return (false);
}
```
Chapter 7 Configuring Conventional Caching Services for Standalone Content Engines

Configuring FTP Caching for Standalone Content Engines

```
label success;
gossl;
return (true);
```

Step 17 Save this firewall script as a file in the default location.

Step 18 From the Tools menu, select **Options > Firewall**.

Step 19 Select the Content Engine that you want to proxy FTP requests through. This is the Content Engine that you specified in **Step 10** of this procedure.

Step 20 Click **Edit**.

Step 21 Select the type field as script `<fswc>` (firescript name).

Tip The firewall script will actually proxy through the Content Engine with the site or username format internally.

Step 22 From the WS-FTP Home window, select **Tools > Site Manager**. The Site Manager window appears. (See Figure 7-9.)

**Figure 7-9 Site Manager Window**

![Site Manager Window](image)

Step 23 In the Site Manager window, click on the site that is created already (for example, “wrq.com”), and click **Edit**. The Site Options window appears. (See Figure 7-10.)
Figure 7-10  Site Options Window

Step 24  In the Site Options window, click Advanced.
Step 25  In the displayed Firewall drop-down list, select the Content Engine that has already been configured.

The following example shows how to use a UNIX command line FTP program to configure the client side proxy FTP request to the Content Engine that is acting as the nontransparent FTP proxy server:

```
shell# ftp -d 10.1.1.50 8501
Connected to 10.1.1.50
220 Welcome to FTP-proxy. Login to the proxy using username and password.
Name (10.1.1.50:admin): smartuser@abchost.company.com
--> USER smartuser
331 Password required for smartuser.
Password:
--> PASS XXXX
220 Welcome to FTP-proxy.
220 Login to origin server using the 'USER username@server-hostname' command, or
220 Login to origin server using the 'SITE server-hostname' followed by the 'USER
username' command.
ftp> site host.abchost.com
--> SITE host.abchost.com
ftp> user anonymous
--> USER anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
--> PASS XXXX
230 Guest login ok, access restrictions apply.
ftp> quit
--> QUIT
shell#
```
Chapter 7 Configuring Conventional Caching Services for Standalone Content Engines

Configuring Transparent FTP Native Caching

The ftp-native service (service 60) is the WCCP caching service that permits WCCP Version 2 routers to redirect transparent FTP native requests transparently to a single port on the Content Engine. The Content Engine retrieves the requested FTP content, stores a copy locally (performs FTP native caching), and serves the requested content to the client.

A standalone Content Engine that is operating as an FTP proxy supports passive and active mode for retrieving files and directories. In FTP native caching mode, if the ftp-native proxy active-mode enable global configuration command is specified, the Content Engine uses the same mode with the FTP server for the data connection as the client used to access the Content Engine, which can be either active or passive:

```
ContentEngine(config)# ftp-native proxy active-mode ?
    enable  Adhere to client's mode for native FTP
```

If the ftp-native proxy active-mode enable command is not specified, the Content Engine uses passive mode with the origin FTP server for the data connection.

If the Content Engine adheres to the client’s mode (active or passive) for native FTP, the following occurs:

- The Content Engine (the FTP native proxy server) performs an active-mode data transfer to or from the origin FTP server if the FTP client issues an active-mode data transfer request.
- The Content Engine performs a passive-mode data transfer to or from the FTP server if the FTP client issues a passive-mode data transfer request.

The format of the URL that the Content Engine creates for a native FTP request depends on the FTP login name and the transfer mode (binary or ASCII file transfer mode):

- If the FTP login name is an actual username instead of “anonymous,” then the string “*user*:password*@” is included in the URL before the host.
- If the mode used to transfer the file is binary mode, then the string “;type=i” is included at the end of the URL. The following is an example of the URL format that the Content Engine creates for a specific user when binary mode is being used:

  ```
  ftp://*user*:password*@10.100.200.5/home/myhome/mybinfile.obj;type=i
  ```

The URL for an “anonymous” user login and an ASCII file transfer mode will not have any fields embedded in the URL, as shown in the following example:

  ```
  ftp://10.100.200.5/home/myhome/mytextfile.txt
  ```

The following two examples demonstrate the use of native FTP with a Content Engine. In the first example, the user logs in with an actual username name (“huff”) and is able to retrieve the requested file (test.c) from the FTP server. In this case, the home directory for the user named “huff” is “/home/huff.”

```
ContentEngine# ftp server.cisco.com
Connected to server.cisco.com.
Name (server:huff): huff
331 Password required for myserver.
Password:
230 User huff logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/home/huff" is current directory.
ftp> get /tmp/test.c
200 PORT command successful.
150 Opening BINARY mode data connection for /tmp/test.c (645 bytes).
```
In the following example, the user logs in as an anonymous user and cannot retrieve the requested file (test.c) because the file is not located in the document root directory of the FTP server ("/"), which is the home directory for any anonymous user:

```
ContentEngine# ftp server.cisco.com
Connected to server.cisco.com.
Name (server:huff): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: test@cisco.com
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/" is current directory.
ftp> passive
Passive mode on
ftp> get (remote-file) /tmp/test.c (local-file) test.c
local: test.c remote: /tmp/test.c
227 Entering Passive Mode (172.31.255.255)
550 /tmp/test.c: No such file or directory.
ftp>
```

To configure transparent FTP native caching with a standalone Content Engine (transparent proxy server) and a WCCP Version 2-enabled router, follow these steps:

**Step 1** Enable FTP native active mode on the Content Engine.
```
ContentEngine(config)# ftp-native proxy active-mode enable
```
In FTP native caching mode, if this command is specified, then the Content Engine uses the same mode (active or passive) with the origin FTP server for the data connection as the client used to access the Content Engine. If this command is not specified, the Content Engine uses passive mode with the origin FTP server for the data connection.

**Step 2** Configure transaction logging on the Content Engine.
```
The following example shows how to configure the Content Engine to use the Apache transaction log format and then enable transaction logging on the Content Engine:
```
ContentEngine(config)# transaction-logs format apache
ContentEngine(config)# transaction-logs enable
```
For more information, see the “Enabling Transaction Logging” section on page 21-33.

**Step 3** Specify the maximum size of an FTP object for FTP native caching. This parameter can be configured for either directory listings or particular objects in the cache:
```
ContentEngine(config)# ftp-native object max-size ?
<1 - 204800> Maximum size of a cacheable object in Kbytes (default is 204800)
```

**Step 4** On the Content Engine, configure transparent FTP native caching.
Transparent redirection of FTP requests is supported only by WCCP Version 2; transparent redirection through a Layer 4 switch is not supported.

a. Define a list of routers that will be used to redirect FTP native requests to this Content Engine. The following example shows how to configure router list 1 to include a single WCCP Version 2 router, the router with the IP address of 10.77.157.41:

```
ContentEngine(config)# wccp router-list 1 10.77.157.41
```

b. Specify the router list that the Content Engine should accept redirected FTP native requests from. The following example shows how to configure the Content Engine to accept redirected FTP native requests for routers that are part of router list 1:

```
ContentEngine(config)# wccp ftp-native router-list-num 1
```

### Step 5
(Optional). Configure the Content Engine to use IP ACLs to grant or deny access to the native FTP proxy service:

```
ContentEngine(config)# ftp-native access-list in {std-acl-num | std-acl-name}
ContentEngine(config)# ftp-native access-list out {ext-acl-num | ext-acl-name}
```

For more information, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.

### Step 6
On the WCCP router, configure the router to transparently intercept FTP native requests and redirect them to the Content Engine that is acting as the FTP proxy server.

a. Enable service 60 on the router.

```
Router(config)# ip wccp 60
```

Service 60 is a predefined WCCP Version 2 caching service that permits WCCP Version 2 enabled routers to redirect FTP native requests transparently to a single port on the Content Engine. The Content Engine retrieves the requested FTP content, stores a copy locally, and serves the requested content to the requester.

b. Specify the interface on which service 60 will run.

```
Router(config)# interface type number
```

c. Configure the router to use the outbound interface for service 60.

```
Router(config-if)# ip wccp 60 redirect out
Router(config-if)# end
```

### Step 7
Verify that the FTP proxy is enabled on the Content Engine:

```
ContentEngine# show wccp modules
```

<table>
<thead>
<tr>
<th>Module</th>
<th>Socket</th>
<th>Expire(sec)</th>
<th>Name</th>
<th>Supported Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>13</td>
<td>4</td>
<td>WMT Proxy</td>
<td>WMT</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>4</td>
<td>MMSU WMT Proxy</td>
<td>MMSU</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>4</td>
<td>WMT-RTSPU</td>
<td>RTSPU</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>4</td>
<td>RTSP Proxy</td>
<td>RTSP</td>
</tr>
<tr>
<td>0</td>
<td>17</td>
<td>3</td>
<td>HTTP Proxy</td>
<td>Web Cache</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reverse Proxy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Custom Web Cache</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HTTPS Cache</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WCCPv2 Service 90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WCCPv2 Service 91</td>
</tr>
</tbody>
</table>
Step 8 Verify that transparent FTP native caching (shown as “FTP” in the following command output) is configured on the Content Engine.

ContentEngine# show wccp services
Services configured on this Content Engine
  Web Cache
  Reverse Proxy
  RTSP
  WMT
  MMSU
  DNS
  FTP
  RTSPU
  HTTPS Cache
ContentEngine#

Step 9 (Optional). Specify a mask used in Content Engine assignments for incoming transparent FTP native requests with the wccp ftp-native mask global configuration command. For example, use the dst-ip-mask command option to set the mask to match the destination IP address of the redirected packet. The destination IP address mask is defined as a hexadecimal number (for example, 0xFC000000). The range is 0x00000000 to FC000000.

ContentEngine(config)# wccp ftp-native mask
  dst-ip-mask Specify sub-mask used in packet destination-IP address
  src-ip-mask Specify sub-mask used in packet source-IP address
ContentEngine(config)# wccp ftp-native mask

For more information about specifying a mask for a WCCP service, see the “Configuring Layer 2 Forwarding with the Mask Load-Balancing Method” section on page 6-36.

Step 10 Enter the l2-redirect option to specify Layer 2 redirection as the packet-forwarding method (as opposed to GRE). Enter the mask-assign option to specify mask assignment as the load-balancing method (as opposed to the default hash assignment method) for the FTP native caching service.

ContentEngine(config)# wccp ftp-native router-list-num 1 l2-redirect mask-assign
WCCP configuration for FTP succeeded.
Please remember to config WCCP service 60 on the corresponding router.
ContentEngine(config)#

Step 11 View the configuration for the masks for transparent FTP native requests.

ContentEngine# show wccp masks ftp-native

Step 12 View the current FTP native proxy configuration.

ContentEngine# show ftp-native
Step 13  Display the statistics for the FTP native requests that this standalone Content Engine has handled.

ContentEngine# show statistics ftp-native

The command output shows the number of FTP native GET requests received by the Content Engine, the number of FTP native hits and misses for GET requests, as well as the number of FTP native PUT requests that have been received by this Content Engine. In the ACNS 5.4.1 software and later releases, the command output also shows the number of transparent FTP native requests, the number of nontransparent FTP native requests, the number of FTP-native proxy authentication requests, and the number of failed FTP-native proxy authentication requests.

Step 14  Clear the FTP native statistics on the Content Engine.

ContentEngine# clear statistics ftp-native

Configuring the TFTP Server and Gateway for Standalone Content Engines

In the ACNS 5.1 software and later releases, the Trivial File Transfer Protocol (TFTP) gateway feature enables Content Engines to serve content files requested by networking devices that use native TFTP. Content Engines now perform TFTP-to-HTTP or TFTP-to-FTP translation, eliminating the need for the system administrator to configure and manage a dedicated TFTP server to serve TFTP requests. This feature allows the Content Engine to accept native TFTP requests from the client at the front end and serve the request using the HTTP or FTP protocol at the back end.

Content files include router software images, router configurations, set top box images, IP phone configuration files, and so forth. If the requested file is not available on the Content Engine, the Content Engine caches the file from the origin server. The Content Engine that is functioning as a caching engine, retrieves the file from the Internet on behalf of the device and forwards it to the device. Future requests by any devices for the same file are satisfied by forwarding the file from the Content Engine’s local cache.

Note

The Content Engine does not support transparently intercepted TFTP requests. Every TFTP server request addressed to the Content Engine must have the Content Engine IP address as its destination address.

After the TFTP server has been enabled on the Content Engine, and a client sends a TFTP request for a file, the following events occur:

1. The TFTP server on the Content Engine checks the access control list that is assigned to the TFTP application for authorization.
2. If the request is authorized, the TFTP server checks the directory specified in the TFTP request for the content file. If the request does not contain any directory path, the server searches the default local directory for the file.
3. If the requested file does not exist in a local directory, the TFTP server on the Content Engine creates an HTTP or FTP URL and sends it to the caching application.
4. The caching application searches for the requested file in the cache file system (cfs) and then in the pre-positioned content (cdnfs). If the file is found, it is sent to the TFTP server on the Content Engine.
5. If the requested file is not found, the caching application requests the file from the origin server specified by the URL, and then caches the content.

6. The cached file is then sent to the TFTP server on the Content Engine, which replies to the TFTP client. If the file is not found, a 404 “File not found” message is sent.

Using the TFTP Service and Gateway on Standalone Content Engines

You can use the `tftp-server` global configuration command to configure the TFTP service and gateway on a standalone Content Engine to serve content in response to TFTP requests in two ways:

- Serve local content—Configure local directories and enable the TFTP server on the Content Engine, as described in the next section, “Enabling the TFTP Server and Gateway.”
- Serve content from remote servers—Configure the TFTP gateway feature on the Content Engine, as described in the next section, “Enabling the TFTP Server and Gateway.”

When both the TFTP server and gateway are enabled, the Content Engine responds to TFTP requests by searching for files in its default local directory if the full pathname is not specified in the request. Otherwise, it looks in the local directory that matches the directory in the pathname. If the file is not found, it uses the HTTP or FTP protocol to forward the request to the Content Engine that is functioning as a caching engine. If the file is found on a remote server, the Content Engine caches the file and sends the file to the client that issued the request. The Content Engine replies to subsequent requests for the file from its local cache. If the file is not found, the Content Engine replies to the request with a 404 “File not found” message.

By default, the TFTP service is disabled and access to the TFTP server is denied.

The default local directory assigned to the TFTP server is `/tftpboot`. However, this directory must be created using the `mkdir` command.

The TFTP timeout value is fixed at 5 seconds, and the number of retries is fixed at five retries. These values are nonconfigurable.

Enabling the TFTP Server and Gateway

To serve requests for local content, follow these steps:

**Step 1**
Enable the TFTP service on the Content Engine.

```
ContentEngine (config)# inetd enable tftp
```

**Step 2**
Configure the local TFTP directories on a standalone Content Engine, as follows:

a. Create the local directories using the `mkdir` command.

b. Identify the local directories to the TFTP server using the `tftp-server dir` command.

When you use the `tftp-server dir` command to identify one or more local directories, the first directory identified becomes the default directory. Enter the `tftp-server dir` command once for each directory that you want to identify.

The TFTP server searches for files without a fully qualified pathname in its default directory. The TFTP server only looks for files in the other local directories if the TFTP request explicitly identifies the directory.
Configuring the TFTP Server and Gateway for Standalone Content Engines

If you do not configure any local directories, /tftpboot is automatically assigned as the default directory. However, you would still need to create the /tftpboot directory using the `mkdir` command before the TFTP server can serve requests.

To use the TFTP gateway to serve content from remote servers, you must enable the TFTP server and also identify the remote servers to which the TFTP gateway (the standalone Content Engine) will direct requests. To identify the servers to which the Content Engine will direct requests when it cannot find the requested files in its local directories, use the `tftp-server gw proto` command. For more information on this topic, see the next section, “Configuring the TFTP Server and Gateway on Standalone Content Engines.”

To enable the TFTP server or gateway on a standalone Content Engine, follow these steps:

**Step 1** Enable the TFTP service on the Content Engine.

```
ContentEngine(config)# inetd enable tftp
```

**Step 2** Define an access list that will permit access to the TFTP service by using the `ip access-list` global configuration command.

**Step 3** Apply the access list to the TFTP service by using the `tftp-server access-list` command.

**Step 4** Use the different options of the `tftp-server` command to configure the TFTP server, as described in the next section.

---

**Note** The Content Engine does not support transparent TFTP requests. It only accepts TFTP requests that explicitly contain the Content Engine hostname or IP address.

**Configuring the TFTP Server and Gateway on Standalone Content Engines**

The TFTP server searches for files in the default local directory when it receives a request that does not identify the full directory path to the file. If you do not configure any local directories, the default directory is /tftpboot. Although this directory is automatically assigned as the default directory, you still need to create it (or any other directories you assign to the TFTP server) using the `mkdir` EXEC command.

When you use the `tftp-server dir` global configuration command to identify one or more local directories, the first directory identified becomes the default directory. Enter the `tftp-server dir` global configuration command once for each directory that you want to identify.

The TFTP server only looks for files in the other directories if the TFTP request explicitly identifies the directory.

To configure the TFTP server and gateway on a standalone Content Engine, use the `tftp-server` global configuration command and follow these steps:

**Step 1** Identify one or more local directories that the Content Engine should search for requested files when the full pathname is not included in the TFTP request with the `tftp-server dir` global configuration command.
For example, the following commands configure two local directories from which the Content Engine will try to fulfill TFTP requests:

```
ContentEngine(config)# mkdir /local/mydir
ContentEngine(config)# mkdir /local/clients
ContentEngine(config)# tftp-server dir /local/mydir
ContentEngine(config)# tftp-server dir /local/clients
```

The first directory specified, /local1/mydir, is considered the default directory.

**Step 2** Identify the IP access control list (ACL) that allows access to the TFTP server and gateway.

```
tftp-server access-list {acl-num | acl-name}
```

For example, configure the Content Engine to check access list 1 to determine if TFTP access should be allowed or denied:

```
ContentEngine(config)# tftp-server access-list 1
```

**Step 3** Enable the TFTP gateway feature and identify specific servers to which TFTP requests will be directed when the Content Engine cannot find the requested files in its local directories.

```
tftp-server gw proto {ftp | http} server {hostname | ip_address} [name name passwd password] [path directory] pri priority
```

**Note** When you enter this command, you identify the protocol (FTP or HTTP), the hostname or IP address of the server, and the authentication information required to access each server. You can enter the `tftp-server gw proto` global configuration command twice to configure a primary and backup HTTP or FTP servers. Use the `priority` option to specify whether the server is primary (priority = 1) or backup (priority = 2).

**Note** Authenticated objects are never cached by the Content Engine for HTTP or FTP. If you want to cache these objects, leave the username and password fields in the `tftp-server gw` command blank. When used with FTP, this configuration is equivalent to allowing anonymous access.

Table 7-13 describes the parameters for the `tftp-server gw proto` global configuration command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gw</td>
<td>Configures TFTP gateway functionality for the Content Engine.</td>
</tr>
<tr>
<td>proto</td>
<td>Configures the protocol used to access the origin server to which the TFTP gateway will forward requests when the file cannot be found in a local directory.</td>
</tr>
<tr>
<td>ftp</td>
<td>Uses the FTP protocol to access the origin server.</td>
</tr>
<tr>
<td>http</td>
<td>Uses the HTTP protocol to access the origin server.</td>
</tr>
<tr>
<td>server</td>
<td>Configures the origin server.</td>
</tr>
<tr>
<td>hostname</td>
<td>Hostname of the origin server.</td>
</tr>
<tr>
<td>ip-address</td>
<td>IP address of the origin server.</td>
</tr>
<tr>
<td>name</td>
<td>(Optional) Sets the username for authentication to the origin server.</td>
</tr>
<tr>
<td>passwd</td>
<td>(Optional) Sets the password for authentication to the origin server.</td>
</tr>
</tbody>
</table>
Step 4 Enable the TFTP server on a standalone Content Engine:

ContentEngine(config)# inetd enable tftp

a. Define a standard access list that permits access to the TFTP service for FTP clients on the 192.168.1.0 subnetwork:

ContentEngine(config)# ip access-list standard 1
ContentEngine(config-std-nacl)# ip access-list permit 192.168.1.0 0.0.0.255
ContentEngine(config-std-nacl)# exit

b. Configure two local directories from which the Content Engine will try to fulfill TFTP requests:

ContentEngine(config)# tftp-server dir /local1/mydir
ContentEngine(config)# tftp-server dir /local1/clients

The first directory specified, /local1/mydir, is considered the default directory.

c. Specify the IP access list that should be used to permit access to the TFTP service:

ContentEngine(config)# tftp-server access-list 1

d. Enable the TFTP gateway feature on a standalone Content Engine, and identify the FTP server to which the Content Engine should forward requests when it cannot find the file in its local directories. Set the username and password for authentication to the origin server.

ContentEngine(config)# tftp-server gw proto ftp 192.168.100.1 pri 1 path /myremotedir name myuser passwd mypassword

The directory name /myremotedir is used in the URL sent by the ACNS caching service on the Content Engine to retrieve the file from the remote server. The URL created by using this sample configuration would be as follows:

ftp://myuser:mypasswd@192.168.100.1/myremotedir/requested-file-name

### Configuring DNS Caching for Standalone Content Engines

This section describes how to deploy Domain Name System (DNS) caching on standalone Content Engines, and covers the following topics:

- **About DNS Caching for Standalone Content Engines, page 7-63**
- **Configuring the DNS Caching Service (Service 53) for Standalone Content Engines, page 7-65**
About DNS Caching for Standalone Content Engines

DNS is a system used in the Internet for translating names of network nodes into IP addresses. DNS allows the network to translate domain names entered in requests into their associated IP addresses. For example, when end users (web clients) enter http://www.cisco.com into their browsers, DNS translates the domain name cisco.com into its associated IP address so that these requests can be processed (that is, the requested content can be served to the web clients).

DNS caching allows the Content Engine to cache DNS entries to avoid multiple WAN accesses for DNS server resolution. When you enable DNS caching on a standalone Content Engine, the Content Engine caches the results of recent DNS queries for faster resolution of identical queries in the future. This cached information is then made available to clients making future requests. The ability to store DNS information that can then be distributed to requesting clients turns the Content Engine into a DNS caching name server.

Caution

It is assumed that you are enabling the DNS caching with WCCP interception on a standalone Content Engine.

In centrally managed ANCS networks, configuring the DNS caching service with WCCP interception on a centrally managed Content Engine causes a conflict with the Content Router, because they will both be listening for DNS requests on the same port (port 53). Consequently, they are mutually exclusive and you should not configure DNS cache support with WCCP interception in such environments. You can, however, enable the standard DNS caching service (without WCCP interception support) in centrally managed ACNS networks.

To configure DNS caching on a standalone Content Engine, you can use the Content Engine GUI or CLI. You must specify the IP address of the DNS server that the Content Engine should use for domain name resolution, and then enable DNS caching on the Content Engine. By default, DNS caching is disabled on a Content Engine.

To enable DNS caching on a standalone Content Engine, you must complete the following tasks:

- Specify the list of DNS servers, which are used by the network to translate requested domain names into IP addresses that the Content Engine should use for domain name resolution.
- Specify the name of the local domain.
- Specify the DNS cache size; that is, the maximum number of records that the DNS cache on the Content Engine should store.
- Enable the WCCP Version 2 DNS caching service (the dns service [service 53]) on the Content Engine.

Transparent interception of DNS requests using WCCP was added in the ACNS 5.1 software release. To enable this feature, you must configure the WCCP Version 2 DNS caching service (service 53) on the Content Engine and the WCCP Version 2-enabled router. For more information on this topic, see the “DNS WCCP Transparent Interception Overview” section on page 7-64.
Domain Name Resolution Requirements

Domain name resolution requires that at least one DNS name server be configured on the Content Engine. You can configure one or more DNS name servers for the Content Engine by defining a list of DNS servers for the Content Engine through the Content Engine GUI (the System > DNS option) or the CLI (the ip name-server global configuration command). For more information about defining this list of DNS servers, see the “Configuring DNS Servers for the DNS Caching Service (Service 53)” section on page 7-64.

DNS WCCP Transparent Interception Overview

For transparent interception of DNS requests using WCCP, you must configure the DNS caching service (service 53) on the Content Engine and on a router that supports WCCP Version 2.

The DNS process interacts with the WCCP process in these ways:

- Maintains the bypass lists.
- Monitors the aliveness of the DNS process to make sure that it can accept requests. If the DNS cache has no servers that are responsive, it will deregister the service until it has acceptable servers.
- Configures and manages the WCCP DNS caching service (the dns service [service 53]).

By default, the ACNS DNS caching service (service 53) uses the DNS servers configured on the Content Engine rather than the original DNS server. For information about how to configure a list of DNS servers on the Content Engine, see the next section, “Configuring DNS Servers for the DNS Caching Service (Service 53).”

Configuring DNS Servers for the DNS Caching Service (Service 53)

By default, the Content Engine uses a DNS server from its list of configured DNS servers for domain name resolution.

- List of configured DNS servers—DNS servers that are used in the network and have been added to the list of DNS servers that the Content Engine should use for domain name resolution. (This list of configured DNS servers is created through the ip name-server command or the System > DNS Content Engine GUI.)
- Original DNS server—DNS server from the original request (hereafter referred to as the original DNS server)

If the DNS WCCP interception feature is enabled (that is, service 53 is configured on the Content Engine and a WCCP Version 2-enabled router), you can use the dns use-original-server global configuration command to define which DNS server a standalone Content Engine should use to resolve a domain name, as described in Table 7-14.
Chapter 7 Configuring Conventional Caching Services for Standalone Content Engines

Configuring DNS Caching for Standalone Content Engines

Table 7-14 Specifying DNS Servers for the DNS Caching Service with WCCP Version 2 Interception

<table>
<thead>
<tr>
<th>CLI Command (Abbreviated Syntax)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>dns use-original-server only</td>
<td>Configures the DNS cache service (service 53) on a Content Engine to use only the original DNS server and not a DNS server from its list of configured DNS servers.</td>
</tr>
<tr>
<td>dns use-original-server after-configured</td>
<td>Configures the DNS cache service on a Content Engine to try the configured DNS servers first and if they fail, then to try the original DNS server.</td>
</tr>
<tr>
<td>dns use-original-server before-configured</td>
<td>Configures the DNS cache service on a Content Engine to try the original DNS server first, then the configured DNS servers.</td>
</tr>
<tr>
<td>no dns use-original-server</td>
<td>Configures the DNS cache service on a Content Engine to use only the list of configured DNS servers. This is the default.</td>
</tr>
</tbody>
</table>

You can use the Content Engine GUI or the CLI to configure one or more DNS servers for the Content Engine.

Note From the Content Engine GUI, choose System > DNS, and use the displayed DNS window. For more information about the DNS window, click the HELP button in the window.

Configuring the DNS Caching Service (Service 53) for Standalone Content Engines

The DNS caching service (the dns service [service 53]) is the WCCP service that permits WCCP Version 2-enabled routers to redirect client requests transparently to a Content Engine for the Content Engine to resolve the DNS name. After the Content Engine resolves the DNS name, it stores the resolved DNS name locally so that it can use these resolved names for future DNS requests.

To configure DNS caching for a standalone Content Engine, follow these steps:

Step 1 Enable WCCP Version 2 on the router. (WCCP Version 1 does not support the dns caching service.)

```
Router# configure terminal
Router(config)# ip wccp version 2
```

Step 2 Enable the dns caching service (service 53) on the router.

```
Router(config)# ip wccp 53
```

Step 3 Specify the interface on which service 53 will run.

```
Router(config)# interface type number
```

Step 4 Configure the router to use the outbound interface for service 53.

```
Router(config-if)# ip wccp 53 redirect out
```
Chapter 7  Configuring Conventional Caching Services for Standalone Content Engines

Configuring DNS Caching for Standalone Content Engines

Step 5  Configure the Content Engine to run WCCP Version 2.

```
ContentEngine(config)# wccp version 2
```

Step 6  Configure the Content Engine to run service 53.

```
ContentEngine(config)# wccp dns
```

Step 7  Configure the DNS server port to listen for new client queries and invoke the query resolution routines. Once the hostname has been resolved to an IP address, it is stored in the memory-based DNS cache.

In the following example, the listener IP address, port number, and hostname are configured first. Then DNS caching is enabled on the Content Engine.

```
ContentEngine(config)# dns listen 10.1.1.0 port 53 hostname acme
```

If the DNS listen name does not match a DNS name, use the `dns pin` global configuration commands to pin an IP address to name mapping. The `dns pin` global configuration commands (both, `cname`, `forward`, and `reverse`) allow you to lock an IP address against a name within the cache. The `forward` option maps the hostname to the IP address. The `reverse` option maps the IP address to the hostname. The both option maps in both the forward and reverse directions. The `cname` option inserts the canonical name (CNAME) mapping.

Step 8  Set the length of time that must elapse before an unanswered request is discarded with the `dns retry-period` global configuration command.

Step 9  Set the interval between retransmission of User Datagram Protocol (UDP) DNS requests sent to an upstream DNS server with the `dns retry-timeout` global configuration command.

Because the DNS protocol is using UDP packets that can be lost or dropped, the burden of retransmitting DNS requests is on the requester. Typically, a retransmit is initiated every 3 seconds until a response is received, or if a response is not received, the request times out after 60 seconds. If a DNS server times out, then a new upstream server is selected to query. If there are no more servers to query upstream, then the initial DNS server contacted returns a DNS failed response to the requesting client.

Step 10  Configure this standalone Content Engine to query the configured name servers repeatedly if the initial DNS server contacted fails to respond with the `dns serial-lookup` global configuration command.

Step 11  Start the DNS server on this standalone Content Engine.

```
ContentEngine(config)# dns enable
```

Note  Enabling the DNS server creates an entry of 127.0.0.1 as the name server for the system and starts the memory-based DNS cache.

Step 12  Specify the maximum number of resource records that can be stored in the DNS cache on this standalone Content Engine with the `dns max-cache-memory` global configuration command. (This is an optional step if you want to use the default setting of 10,000 records.)

To avoid unduly straining overall system resources, it is important that you impose a strict maximum memory limit for Content Engine. In the following example, the size of the DNS cache is set to 20,000 records:

```
ContentEngine(config)# dns-cache size 20000
```
Disabling DNS Caching on Standalone Content Engines

To disable DNS caching on a standalone Content Engine, enter the `no dns-cache size` global configuration command.

```
ContentEngine(config)# no dns-cache size
```

Configuring Standalone Content Engines to Send out TCP Keepalives

By default, the Content Engine does not automatically send out keepalives. To configure a standalone Content Engine to send out TCP keepalives on HTTP connections, you must enter the `http tcp-keepalive enable` global configuration command. After entering the `http tcp-keepalive enable` command, the Content Engine will send out a keepalive every 75 seconds on an HTTP connection. If a response is received, the Content Engine continues to send a keepalive every 75 seconds. If a response if not received (the device does not respond), the Content Engine waits 90 seconds and logs a miss. After four misses, the Content Engine considers the HTTP connection to be down and closes the connection.

Specify how many times the Content Engine should attempt to connect to the device before closing the connection with the `tcp keepalive-probe-cnt` global configuration command. The count can be from 1 to 10. The default is 4 attempts.

Specify how often the Content Engine is to send out a TCP keepalive with the `tcp keepalive-probe-interval` global configuration command. The interval can be from 1 to 120 seconds. The default is 75 seconds.

Configure the Content Engine to wait for a response (the device does not respond) before it logs a miss with the `tcp keepalive-timeout` global configuration command. The timeout can be from 1 to 120 seconds. The default is 90 seconds.
Configuring Persistent Connections on Standalone Content Engine

Content Engines by default use persistent connections to the server for improving performance. In the ACNS 5.0.7 software and later releases, the rule action no-persistent-connection global configuration command allows you to disable or enable persistent connections for specific domains, source and destination IP addresses, or ports. This is useful when a server does not support persistent connections.

Enabling Persistent Connections

The Content Engine keeps a connection persistent if persistence is allowed for the persistence idle timeout period (which is 600 seconds by default). If HTTP persistent connections are enabled, then no keepalive is needed and the Content Engine will keep the connection open until the idle timeout period is exceeded.

*Note* The Content Engine does not automatically send out keepalives. To configure the Content Engine to sent out TCP keepalives over an HTTP connection, you must enter the `http tcp-keepalives enable` global configuration command.

Once a response or data is sent over the persistent connection, the idle period restarts. HTTP persistent connections can be configured for either the client or server or both.

The Content Engine GUI or the CLI can be used to enable and disable persistent connections on standalone Content Engines.

To use the Content Engine GUI to enable persistent connections on a standalone Content Engine, choose Caching > Persist. Connect, and use the displayed Persistent Connections window. For more information about how to use the Persistent Connections window to enable persistent connections, click the HELP button in the window.

To use the Content Engine CLI to enable persistent connections on standalone Content Engines, use the `http persistent-connections` global configuration command:

```bash
ContentEngine(config)# http persistent-connections [all|client-only|server-only|timeout seconds]
```

Table 7-15 describes the HTTP persistent connection parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>persistent-connections</td>
<td>Sets persistent connections configuration options.</td>
</tr>
<tr>
<td>all</td>
<td>(Optional) Makes client and server connections persistent.</td>
</tr>
<tr>
<td>client-only</td>
<td>(Optional) Makes only a client connection persistent.</td>
</tr>
<tr>
<td>server-only</td>
<td>(Optional) Makes only a server connection persistent.</td>
</tr>
<tr>
<td>timeout</td>
<td>(Optional) Sets persistent connections timeout value (idle timeout period).</td>
</tr>
<tr>
<td>seconds</td>
<td>Persistent connections timeout in seconds (1–86400).</td>
</tr>
</tbody>
</table>
The persistence does not start until an initial request is made (for example, a GET request) or until data starts to flow over the persistent connection. If there is no initial request or data sent over a persistent connection, the read-write (rw)-timeout setting takes effect. The rw-timeout setting also is used if the connection goes idle for some reason before it has finished sending or receiving the data. In this case, the connection is timed out for the period specified by the rw-timeout setting. The rw-timeout setting can be set for reading and writing data to either the server or the client through the `tcp server-rw-timeout` and `tcp client-rw-timeout` global configuration commands. By default, the rw-timeout for both the server and the client is set to 120 seconds. For more information on this topic, see the “Viewing or Modifying TCP Parameters on Standalone Content Engines” section on page 20-2.

### Disabling Persistent Connections

To disable all persistent connections, client-only persistent connections, or server-only persistent connections on a standalone Content Engine use the no form of the `http persistent-connections [all | client-only | server-only | timeout seconds]` global configuration command:

```
ContentEngine(config)# no http persistent-connections [all|client-only|server-only|timeout seconds]
```

To disable specific persistent connections to specific domains, IP addresses, or ports use the `rule action no-persistent-connection` global configuration command:

```
ContentEngine(config)# rule action no-persistent-connection
pattern-list list_num [protocol {http|https|ftp}]
```

The `rule action no-persistent-connection` global configuration command has the following options:

- **all**—Do not use persistent connection for all connections.
- **client-only**—Do not use persistent connection for client connections only.
- **server-only**—Do not use persistent connection for server connections only.

You can specify the criteria for disabling persistent connections by creating a pattern list using one or more of the following supported patterns:

- `src-ip`
- `dst-ip`
- `dst-port`
- `url-regex`
- `domain`
- `header-field user-agent`
- `header-field referer`
- `header-field request-line`

Table 7-16 describes the syntax for the `rule action no-persistent-connection` command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>Describes the action that the rule is to take.</td>
</tr>
<tr>
<td>no-persistent-connection</td>
<td>Sets persistent connection configuration options.</td>
</tr>
<tr>
<td>pattern-list</td>
<td>Configures the pattern list.</td>
</tr>
</tbody>
</table>
Configuring Healing Mode for Content Engine Clusters

Healing mode allows a newly added Content Engine to query and obtain cache objects from all other Content Engines in the Content Engine cluster on a cache miss. If the object is not found in the cluster, the Content Engine processes the request through the outgoing proxy or origin server. The Content Engine in healing mode is called a healing client. The Content Engines in the Content Engine cluster that respond to healing client requests are called healing servers.

When a Content Engine is added to an existing Content Engine cluster running WCCP Version 2, it can receive requests for content that was formerly served by another Content Engine in the Content Engine 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.

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 client sends the request directly to the Content Engine (acting as a nontransparent forward proxy server).

To allow a Content Engine in a Content Engine farm to query and obtain cache objects from other Content Engines in the cluster, you must enable healing mode on the Content Engine, using the Content Engine GUI or CLI to enable healing mode on a standalone Content Engine.

Table 7-16 Parameters of the rule action no-persistent-connection Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list_num</td>
<td>Pattern list number (1 to 512).</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol for which this rule is to be matched.</td>
</tr>
<tr>
<td>http</td>
<td>Matches this rule with the HTTP protocol.</td>
</tr>
<tr>
<td>https</td>
<td>Matches this rule with the HTTPS protocol.</td>
</tr>
<tr>
<td>ftp</td>
<td>Matches this rule with the FTP protocol.</td>
</tr>
</tbody>
</table>

The following example disables a persistent connection for the domain mywebsite.com, based on a pattern in pattern list 10:

```
ContentEngine(config)# rule action no-persistent-connection server-only pattern-list 10
WARNING: rule action no-persistent-connection will affect end-to-end NTLM authentication to these servers
ContentEngine(config)#
```

```
ContentEngine# show rule all
Rules Template Configuration
-------------------------------
Rule Processing Enabled
Actions :
rule action no-persistent-connection server-only pattern-list 100 protocol all

Pattern-Lists :
rule pattern-list 100 domain mywebsite.com
ContentEngine#
```

For more information about using the Rules Template feature to configure rules for standalone Content Engines, see Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”
To configure the clustering parameters (the parameters related to WCCP service clusters) from the Content Engine GUI, choose **WCCP > Clustering**. Use the Clustering window to specify these parameters for this Content Engine. For more information about how to use the Clustering window to specify these parameters, click the **HELP** button in the window.

To use the Content Engine CLI to enable healing mode on standalone Content Engines, use the **http cluster** global configuration command:

```
http cluster {heal-port number | http-port number | max-delay seconds | misses number}
```

Table 7-17 describes the **http cluster** command parameters.

**Table 7-17 Healing Mode CLI Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster</td>
<td>Configures cache cluster options for the Content Engine.</td>
</tr>
<tr>
<td>heal-port</td>
<td>Listening port for the healing server for healing requests.</td>
</tr>
<tr>
<td>number</td>
<td>Healing server listener port number (1–65535). The default is 14333.</td>
</tr>
<tr>
<td>http-port</td>
<td>HTTP port number over which requests from the healing Content Engine are sent to other Content Engines in the cluster. The default port number for the HTTP healing port is port 80. Valid port numbers are from 1 to 65535.</td>
</tr>
<tr>
<td>number</td>
<td>HTTP request forwarding port number (1–65535). The default is 80.</td>
</tr>
<tr>
<td>max-delay</td>
<td>Maximum wait for a response from the Content Engine cluster.</td>
</tr>
<tr>
<td>seconds</td>
<td>Maximum delay in seconds (0–10).</td>
</tr>
<tr>
<td>misses</td>
<td>Duration of healing mode for the Content Engine.</td>
</tr>
<tr>
<td>number</td>
<td>Total number of misses (0–999) before healing mode is disabled.</td>
</tr>
</tbody>
</table>

Specify the port number over which requests from the healing Content Engine (healing client) are sent to the healing servers (other Content Engines in the cluster) with the **http cluster http-port** global configuration command. The default port number is port 80. If you choose to configure a port other than port 80, make sure that it matches the port that was specified on the healing servers in the cluster, using the **http proxy incoming** global configuration command. Otherwise, the healing client is not able to retrieve objects from the healing servers.

Specify the port number over which the healing client sends healing queries and the healing server sends healing responses with the **http cluster heal-port** global configuration command. The default port number is 14333. If a port other than the default is configured, make sure that all Content Engines in the cluster use the same port.

Specify the maximum number of misses that the healing Content Engine can receive from the cluster from the last healing mode hit response until the healing process is disabled with the **http cluster misses** global configuration command. The default is 0 misses.

After a WCCP bucket redistribution, the Content Engine will try to populate its cache from other Content Engines on every cache miss. You can configure the maximum number of seconds a Content Engine should wait for a response from its neighbors before retrieving the object itself. The default is 0 seconds. To specify the maximum time in seconds that a healing Content Engine waits for a healing response from the cluster before considering the healing request a miss, use the **http cluster max-delay** global configuration command.

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. If you use the default port, you do not have to configure **http-port**. The default port number for **heal-port** is 14333.
Chapter 7  Configuring Conventional Caching Services for Standalone Content Engines

Configuring the Internet Cache Protocol for Content Engine Clusters

Internet Cache Protocol (ICP) is a lightweight message format used for communicating among Content Engines and for supporting interoperability with older proxy protocols. ICP is used to exchange hints about the existence of URLs in neighboring Content Engines in a Content Engine cluster (farm). Content Engines 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 been used traditionally to scale the overall size of a cluster of Content Engines beyond a single unit, history has shown ICP to be a poor way of scaling a Content Engine clustering solution. In fact, because of the way that traffic is currently directed toward a transparent network Content Engine cluster, the requirement for ICP is all but negated for the majority of Content Engine 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

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

The following example shows how to use the Content Engine CLI to restrict the ICP parent and sibling to specific domain sets:

```
ContentEngine(config)# icp client add-remote-server 10.1.1.1 parent icp-port 3130 http-port 3128 domain_x.com domain_y.com domain_z.com
ContentEngine(config)# icp client add-remote-server 10.1.1.1 sibling icp-port 3130 http-port 3128 domain_a.com domain_b.com domain_c.com
ContentEngine(config)# icp client enable
```

You can use the Content Engine CLI or GUI to configure ICP on a standalone Content Engine that is part of a cache cluster, as described in the following sections:

- Configuring Standalone Content Engines as ICP Clients, page 7-73
- Configuring Standalone Content Engines as ICP Servers, page 7-74
Configuring Standalone Content Engines as ICP Clients

You can configure your Content Engine cluster to generate ICP queries before retrieving requested objects from the Internet using ICP client functionality. With ICP, you can configure parent and sibling Content Engines in a caching hierarchy. ICP parents are essentially one step higher than ICP siblings in a hierarchy of Content Engines.

You can configure a standalone Content Engine to be either a parent or a sibling:

- Parent Content Engines are able to retrieve data during a cache miss.
- Sibling Content Engines cannot retrieve data and instead forward the request to the parent Content Engines.

You can use the Content Engine CLI or GUI to configure a standalone Content Engine as an ICP client, as follows:

- From the Content Engine GUI, choose **Caching > ICP Client**, and use the ICP Client window. To obtain more information about this window, click the **HELP** button in the window.
- From the Content Engine CLI, use the `icp client` global configuration commands to configure a standalone Content Engine as an ICP client. Configurations made without enabling ICP functionality are stored within the configuration until removed.

Table **7-18** describes the parameters of the `icp client` global configuration command.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>icp client enable</td>
<td>Enables the ICP client on a Content Engine.</td>
</tr>
<tr>
<td>icp client add-remote-server</td>
<td>Adds a remote ICP client server.</td>
</tr>
<tr>
<td>icp client exclude</td>
<td>Excludes ICP client local domains.</td>
</tr>
<tr>
<td>icp client max-fail</td>
<td>Sets the maximum number of retries allowed. Valid values are 0 to 100. The default is 20.</td>
</tr>
<tr>
<td>icp client max-wait</td>
<td>Configures how long the Content Engine waits before retrieving the requested data directly from the Internet.</td>
</tr>
<tr>
<td>icp client modify-remote-server</td>
<td>Modifies the ICP client remote server parameters.</td>
</tr>
</tbody>
</table>

The following example shows how to use the Content Engine CLI to restrict ICP parent and sibling to specific domain sets:

```
ContentEngine(config)# icp client add-remote-server 172.16.0.0 parent icp-port 3130 http-port 3128 domain_x.com domain_y.com domain_z.com
ContentEngine(config)# icp client add-remote-server 172.16.0.0 sibling icp-port 3130 http-port 3128 domain_a.com domain_b.com domain_c.com
ContentEngine(config)# icp client enable
Icp Client started
```
Configuring Standalone Content Engines as ICP Servers

You can also configure a standalone Content Engine to act as an ICP server. This allows the Content Engine to probe the hierarchy of Content Engines by multicasting an ICP message to ICP parent and sibling clients in the hierarchy.

You can use the Content Engine GUI or the CLI to configure a standalone Content Engine as an ICP server, as follows:

- From the Content Engine GUI, choose Caching > ICP Server and use the displayed ICP Server window. To obtain more information about this window, click the HELP button in the window.
- From the Content Engine CLI, use the icp server global configuration commands to establish and configure the Content Engine as an ICP server. Configurations made without enabling ICP functionality are stored within the configuration until removed.

Table 7-19 describes the parameters for the icp server global configuration command.

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>icp server enable</td>
<td>Enables the ICP server on a Content Engine.</td>
</tr>
<tr>
<td>icp server http-port</td>
<td>Configures the HTTP proxy port on a Content Engine to listen for ICP-generated requests. The range is from 0 to 65535. The default port number is 3128.</td>
</tr>
<tr>
<td>icp server port</td>
<td>Configures the ICP server port on a Content Engine to listen for ICP requests. The range is from 0 to 65535. The default port number is 3130.</td>
</tr>
</tbody>
</table>
Chapter 8

Configuring RealMedia Services on Standalone Content Engines

This chapter describes how to configure RealMedia streaming and caching services on standalone Content Engines, including how to configure the Real-Time Streaming Protocol (RTSP) gateway that runs on the Content Engine. For information about how to configure Windows Media Technologies (WMT) streaming and caching services (WMT RTSP streaming and caching services) on standalone Content Engines, see Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”

This chapter contains the following sections:

- Overview of the RealMedia Streaming Solution, page 8-2
- Configuring RealMedia Services, page 8-9
- Verifying RealProxy Configurations for Standalone Content Engines, page 8-23
- Restoring the RealProxy Factory-Default Configuration on Standalone Content Engines, page 8-29
- Restarting RealProxy on Standalone Content Engines, page 8-30
- Disabling RealMedia Caching on Standalone Content Engines, page 8-30
- Uninstalling the RealProxy License Key, page 8-31
- Displaying RealProxy Statistics for Standalone Content Engines, page 8-31

For background information about streaming media services, see the “Understanding Some Basic ACNS Streaming Media Concepts” section on page 2-10. For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication. For information about how to configure streaming media services for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

Note

All cached RealMedia content is deleted from the Content Engine mediafs cache when you upgrade a Content Engine from any ACNS software release to the ACNS 5.1.9 software and later releases, or the ACNS 5.2.1 software and later releases. This deletion occurs because the meta file formats have been changed in these releases, affecting the way that the cached RealMedia streaming file is interpreted.
Overview of the RealMedia Streaming Solution

RealMedia is the streaming media solution from RealNetworks, Inc. RealMedia uses the RealNetworks RTSP protocol (IETF standard RTSP protocol plus proprietary extensions) to deliver streaming media content to RealMedia clients (for example, RealPlayer 8.0 and RealOne players).

RealMedia has two main software components: RealProxy and RealSubscriber. With registered Content Engines, you can enable and run RealProxy and RealSubscriber; however, standalone Content Engines only support RealProxy. You cannot enable and run RealSubscriber on a standalone Content Engine.

The RealProxy software from RealNetworks, Inc., included as a software option in the ACNS 5.x software allows you to deploy the following RealMedia services on standalone Content Engines:

- Live splitting (distributing live feeds)
- Streaming video-on-demand (VOD) files in an RTSP-based format
- Caching VOD files

Table 8-1 describes RealMedia supported services for standalone Content Engines.

Table 8-1  RealMedia Streaming and Caching Services Supported for Standalone Content Engines

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealMedia proxy caching of VOD files</td>
<td>The standalone Content Engine is functioning as a nontransparent proxy server for end users who are using a RealMedia player to request content. After receiving an RTSP request directly from a RealMedia player, the Content Engine retrieves the requested VOD file if it is not already stored in its local cache, stores a copy locally, and sends the requested content to the RealMedia player. See the “Configuring Direct Proxy Routing and RealMedia Proxy Caching” section on page 8-21.</td>
</tr>
<tr>
<td>RealMedia transparent caching of VOD files</td>
<td>The standalone Content Engine is functioning as a transparent proxy server for end users who are using a RealMedia player to request content. After receiving a redirected RTSP request from a WCCP Version 2 router or Layer 4 switch, the Content Engine retrieves the requested content, stores a copy locally, and sends the requested content to the RealMedia player. See the “Configuring RTSP Transparent Redirection and Caching of RealMedia Requests” section on page 8-17.</td>
</tr>
<tr>
<td>RealProxy live splitting</td>
<td>The Content Engine serves RTSP live streams to all local users (RealMedia players) whose requests are directed to it. The RTSP live streams can be unicast live feeds or multicast live feeds. The Content Engine splits the live feeds into a multicast or unicast to relay the stream to the RealMedia client. Live streams are not files so they cannot be cached but VOD files can be cached. See the “About Live Splitting and Caching VOD Files with RealProxy” section on page 8-6.</td>
</tr>
</tbody>
</table>

Note  In the ACNS 5.2.1 software release, support for Synchronized Multimedia Integration Language (SMIL) files in RealProxy was added. For more information, see the “About RealProxy SMIL File Support” section on page 8-8.
Table 8-2 describes the RealProxy features and benefits for standalone Content Engines.

<table>
<thead>
<tr>
<th>RealProxy Feature</th>
<th>Description</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy for RealMedia players (for example, RealPlayer 8.0 or RealOne players)</td>
<td>RealProxy makes requests for content on behalf of the RealMedia clients.</td>
<td>Manages traffic inside the firewall by coordinating requests for similar content. Masks end user IP addresses.</td>
</tr>
<tr>
<td>Splitting support for live broadcasts</td>
<td>RealProxy splits a single inbound live broadcast feed to multiple RealMedia clients.</td>
<td>Reduces inbound bandwidth usage to a single stream of content during a live event.</td>
</tr>
<tr>
<td>Caching of RealSystem G2 and Progressive Network Audio (PNA) content</td>
<td>RealProxy caches all proxied streaming media traffic from RealNetworks servers. RealProxy caches content locally after authentication with the origin streaming server.</td>
<td>Significantly reduces inbound bandwidth usage by eliminating redundant file transmissions across the network.</td>
</tr>
<tr>
<td>Authentication and accounting</td>
<td>RealProxy authenticates every content request with the origin streaming server before delivering the cached content to the clients.</td>
<td>Retains access to general usage data for the broadcaster. Appropriately authenticates users. Guarantees the freshest content for end users.</td>
</tr>
<tr>
<td>Aggregate bandwidth thresholds</td>
<td>RealProxy thresholds allow you to specify the maximum bandwidth for inbound and outbound RTSP traffic (cached content and live content).</td>
<td>Provides control over aggregate bandwidth usage within the network and prevents stress on mission-critical applications.</td>
</tr>
<tr>
<td>Proxy routing</td>
<td>RealProxy can 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.</td>
<td>Allows you to proxy route requests, providing an additional level of control.</td>
</tr>
</tbody>
</table>

The Content Engine can be configured to accept transparently redirected content requests as well as traditional proxy-style requests from RealMedia players. The redirection of RTSP traffic to the Content Engine media cache is enabled through the Content Engine GUI or CLI. The RealProxy software is configured with the RealSystem administrator GUI, accessed from the RealProxy window of the Content Engine GUI. (See the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21.)

The RealProxy feature on a standalone Content Engine is licensed software. For more information on this topic, see the “About the RealProxy License Key” section on page 8-9.
About the RTSP Gateway and Backend RTSP Servers

The RTSP gateway is the single point of entry for RTSP messages. The RTSP gateway is automatically enabled and runs on the standalone Content Engine. The RTSP gateway listens on the standard RTSP port (default port 554) and funnels incoming RTSP traffic through it to enforce rules-based implementation and URL filtering of RTSP requests.

For every RTSP request, the RTSP gateway examines the following properties of the request:

- The URL and its position in the UNS
- The user agent
- The IP address of the final destination
- The media type

The possible actions that the RTSP gateway can take on an RTSP streaming request are as follows:

- Tunnels the request to the RealProxy server, an RTSP backend server that is running on the same Content Engine as the RTSP gateway.
- Transfers the socket fd to the Windows Media server that is running on the same Content Engine as the RTSP gateway.

If the user-agent is Windows Media Player, then instead of tunneling the RTSP request to the Windows Media server (the mms_server process), the RTSP gateway transfers the socket fd (kick fd) to the Windows Media server that is running on the Content Engine. This method allows Windows Media clients to communicate directly with the Windows Media server for subsequent RTSP requests, which eliminates the RTSP gateway process from subsequent RTSP request flows.

- Redirects the request

The RTSP gateway checks if the request matches any of the configured rules. Rules may decide if a particular request does not need to go through authentication and authorization. If the RTSP request is successfully authenticated and authorized, URL filtering is performed on the request. If the request is a WCCP-transparently redirected request and the Content Engine cannot handle the redirect, the RTSP gateway redirects the request and adds bypass entries.

- Rejects the request

Note: The RTSP gateway supports local list URL filtering (good site and bad site lists) for RTSP requests. URL filtering with third-party software (for example, SmartFilter, Websense, and N2H2) is not supported for RTSP requests.

If the RTSP request is blocked as a result of URL filtering, then the client receives a 403 Forbidden error message. Otherwise, the RTSP gateway sends the RTSP request to the appropriate backend RTSP server based on the properties of the incoming request, including such properties as the client player, final destination, and media file type. The RTSP gateway encodes the URL so that the backend RTSP servers know the original URL before unified name space (UNS) translation.

On standalone Content Engines that are running the ACNS 5.2 software and earlier releases, the RealProxy server is the only backend RTSP server that can be enabled on the Content Engine. In the ACNS 5.3.1 software and later releases, you can enable the RealProxy server or the WMT RTSP server as backend RTSP servers on standalone Content Engines.
Table 8-3 lists the supported backend RTSP servers for standalone Content Engines that are running the ACNS 5.3.1 software and later releases.

### Table 8-3  Backend RTSP Servers Supported on Standalone Content Engines

<table>
<thead>
<tr>
<th>RTSP Streaming Solution</th>
<th>RTSP Backend Servers</th>
<th>RTSP-Based Clients</th>
<th>Protocol Used to Service RTSP-based Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealMedia from RealNetworks, Inc.</td>
<td>RealProxy server</td>
<td>RealMedia players</td>
<td>RealProxy server uses RealNetworks RTSP proprietary protocol over RTP to stream RTSP content to RealMedia players.</td>
</tr>
<tr>
<td>Windows Media 9 Series from Microsoft Corporation</td>
<td>Windows Media 9 RTSP server (WMT RTSP server)</td>
<td>Windows Media 9 players</td>
<td>WMT RTSP server uses the IETF standard RTSP protocol (plus proprietary Microsoft extensions) to stream content to Windows Media 9 players.</td>
</tr>
</tbody>
</table>

Note

For Content Engines that are registered with a Content Distribution Manager, you can also enable RealSubscriber and Cisco Streaming Engine as backend RTSP servers that run on the Content Engine. For information about how to configure Cisco Streaming Engine and RealSubscriber for registered Content Engines, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

The following sample output from the `show rtsp all` EXEC command shows a sample configuration for Content Engine 24, a standalone Content Engine that is has been configured to use the RTSP gateway and two RTSP backend servers (the WMT RTSP server and the RealProxy server) to service RTSP requests from RealMedia and Windows Media 9 players:

```
ContentEngine 24# show rtsp all

RTSP Gateway Configuration
---------------------------
RTSP Gateway ip-address 209.165.202.128
RTSP Gateway incoming port 554
RTSP Gateway incoming request rate limit is 40 requests/sec
RTSP Gateway initial setup delay is 10 sec
RTSP Gateway L4-switch is enabled
RTSP Gateway Transparent Interception (WCCP):
  Not configured.

WMT RTSP Server/Proxy Configuration
------------------------------------
WMT version: ce507-001.000
WMT license key is installed
WMT evaluation is not enabled
WMT end user license agreement accepted
WMT is enabled
WMT disallowed client protocols: none
WMT outgoing bandwidth configured is 1 Kbits/sec
WMT incoming bandwidth configured is 56000 Kbits/sec
WMT max sessions configured: 2500
WMT max sessions platform limit: 2500
WMT max sessions enforced: 2500 sessions
WMT max outgoing bit rate allowed per stream: 2 Kbits/sec
WMT max incoming bit rate allowed per stream: 3 Kbits/sec
WMT debug level: 0
WMT L4 switch is enabled
WMT debug client ip not set
```
Overview of the RealMedia Streaming Solution

WMT debug server ip not set
WMT fast-start is enabled
WMT fast-start max. bandwidth per player is 65 (Kbps)
WMT fast-cache is enabled
WMT fast-cache acceleration factor is 5
WMT Extended Transaction Log is not enabled
WMT Transaction Log format is Windows Media Services 4.1 logging

Real Proxy Configurations
-------------------------------
Real Proxy version: ce507-9.0.2.794
Real Proxy is not enabled
Real Proxy evaluation is not enabled
Real Proxy license key not installed
Real Proxy end user license agreement accepted
Real Proxy is configured to use 30% of MEDIAFS partition
Real Proxy incoming bandwidth enforced is 0 kbps
Real Proxy outgoing bandwidth enforced is 0 kbps

For registered Content Engines, the command output from the **show rtsp server** EXEC command would also include configuration information about RealSubscriber and the Cisco Streaming Engine (two additional RTSP backend servers that can be enabled on a registered Content Engine).

Each backend RTSP server that is running on the Content Engine performs its own transaction logging. For example, RealProxy uses RealProxy transaction logs while the WMT RTSP server uses the WMT transaction logs. By default, transaction logging is enabled on a Content Engine.

For more information about the RTSP gateway, see the “Configuring the RTSP Gateway for Standalone Content Engines” section on page 8-14. For information about how to configure the WMT RTSP server and WMT RTSP services, see the “Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines” section on page 9-14.

### About Live Splitting and Caching VOD Files with RealProxy

When RealProxy is enabled on a standalone Content Engine, the Content Engine will serve as the stream splitting point for all local users (RealMedia players) whose RealMedia requests is directed to the Content Engine. All subsequent requests to that origin streaming server (the Helix Universal Server) are served by the Content Engine, which splits the stream and serves it to the RealMedia clients.

Figure 8-1 shows how the Content Engine splits one unicast live stream into multiple unicast streams in the local network. When the first client (Client 1) that requested the original stream disconnects from the network, the Content Engine continues to serve the other clients (Client 2 and Client 3) until all clients disconnect from the network.
By having the Content Engine perform the live splitting, you potentially save considerable network bandwidth between the client and the origin streaming server, because the Content Engine is closer to the clients.

**Note**
RealProxy cannot cache live broadcasts (also referred to as *live clips* and *live streams*) because there is no actual downloadable file to cache. However, RealProxy supports live splitting in order to allow RealMedia clients to share the broadcast, which saves WAN bandwidth.

When the Content Engine receives a request for a RealMedia VOD file (.rm file) that is not already stored in its local cache, the Content Engine retrieves the requested VOD file from the origin streaming server (the Helix Universal Server), caches the VOD file if RealMedia caching is enabled on the Content Engine, and streams the requested VOD file to the RealMedia client.

In both operations (streaming of VOD files and live splitting), the RealMedia request can be directed to the Content Engine through one of two methods:

- Transparent redirection through WCCP Version 2 or Layer 4 switching
- Direct proxy routing (explicitly configuring the RealMedia player proxy settings)

Configuration for both of these operations is the same on the Content Engine and routers; the only difference is the source (a VOD file or a live stream).

All incoming RTSP requests are directed to the RTSP gateway on the Content Engine. The RTSP gateway decides which backend RTSP server on the Content Engine (for example, RealProxy) to direct the request to. For more information about the RTSP gateway, see the “Configuring the RTSP Gateway for Standalone Content Engines” section on page 8-14.

For more information about RealMedia caching for standalone Content Engines, see the “Configuring RealMedia Services” section on page 8-9.
Overview of the RealMedia Streaming Solution

Chapter 8 Configuring RealMedia Services on Standalone Content Engines

About Caching Policies in RealMedia Caching

In contrast to HTTP caching, caching policies in RealMedia caching are much simpler, because streaming media are mostly large static content. All responses other than live streams are cacheable (VOD files can be cached), including partial responses. All RealMedia requests result in communication between the Content Engine and the origin streaming server (the Helix Universal Server), even if the request is a cache hit.

By establishing the streaming control session, the Content Engine can verify that its cached content is fresh, and the client can access the content. Because streaming objects are typically very large in size, the overhead of establishing the control session with the server is minimal and does not reduce the bandwidth savings from the cache hits.

Note
Live streams are not cached.

About Access Control

If a RealMedia client requests a cached stream, before the client is allowed to play the stream, the RealProxy server that is running on the Content Engine uses the RealNetworks proprietary RTSP protocol to send the request to the origin streaming server (Helix Universal Server) for permission. If the origin streaming server denies the request, the RealMedia client is not allowed to receive the requested stream.

About RealProxy SMIL File Support

SMIL is a simple but powerful markup language that allows you to coordinate multiple clips. SMIL also allows you to define how, when, and where you want the multiple clips to be played.

A client browser can automatically launch the media player on the client desktop when the video and presentation material is packaged in a media-index file such as an SMIL file. The browser is typically configured so that the moment it retrieves an SMIL file, it automatically launches the RealMedia player on the client desktop, and passes the SMIL file to the client RealMedia player. Media-index files can contain either relative links to media files or absolute links to media files.

Note
The .asx file is used by the WMT streaming solution, and the SMIL file is used by the RealNetwork streaming solution. For more background information on media-index files, see the “How a Client Media Player Issues a Request” section on page 2-14.

SMIL files are used for the following main purposes:
- To describe the overall layout of the SMIL-based presentation.
- To act as the macro meta file for the SMIL-based presentation.

Rather than encoding presentations in a single file, SMIL allows content creators to encode pieces of the presentation in separate files and then use SMIL to control the interaction of these separate files. The SMIL file points to the source of the media and data files, as well as the source of the more specific meta files that comprise the SMIL-based presentation.
- To establish the overall timeline of the SMIL-based presentation.
With a SMIL-based presentation, each element in the presentation can be encoded and transmitted separately, with synchronization control. Consequently, content creators can optimize the delivery of their SMIL-based presentations by specifying the least bandwidth-intensive format for transmittal. In addition to reducing the bandwidth requirements, SMIL also expedites the process of editing an SMIL-based presentation after the presentation is completed. For example, a content creator can use SMIL to easily delay the audio track of a completed SMIL-based presentation (for example, have the audio not start for 5 seconds after the presentation begins) without having to edit the actual audio file.

In the ACNS 5.2.1 software and later releases, SMIL file support for RealProxy is supported. SMIL support is provided under the following conditions:

- **Case 1**—The SMIL file and its contents are pre-positioned on a registered Content Engine.
- **Case 2**—The SMIL file and its contents are not pre-positioned on the Content Engine.
- **Case 3**—The SMIL files have absolute URLs, and each URL is pointing to a different server.

All of the three cases are supported on Content Engines that meet the following requirements:

- The Content Engine is running the ACNS 5.2 software and later releases.
- RealProxy is enabled on the Content Engine.
- The Content Engine is registered with a Content Distribution Manager.

Only Cases 2 and 3 are supported on standalone Content Engines (Content Engines that are not registered with a Content Distribution Manager) that meet the following requirements:

- The Content Engine is running the ACNS 5.2.1 software and later releases.
- RealProxy is enabled on the Content Engine.

### About the RealProxy License Key

The RealNetworks, Inc. RealProxy product is licensed software. To activate the licensed RealProxy feature on a standalone Content Engine, you must have a RealProxy license key. You must specify a permanent RealProxy license key that is supplied on a certificate shipped with the Content Engine, or use an evaluation key for a temporary period. If you are downloading the ACNS 5.x software, you can purchase a RealProxy license through the Cisco.com website. You specify the RealProxy license key as part of enabling the RealProxy feature on a standalone Content Engine. See the “Enabling RealProxy on Standalone Content Engines” section on page 8-13.

### Configuring RealMedia Services

The Content Engine can be configured to accept transparently redirected RTSP requests as well as traditional proxy-style RTSP requests from RealMedia players. RealProxy also supports live splitting and caching of RealMedia VOD files (.rm files).

The Setup utility allows you to enable the licensed RealProxy feature on a standalone Content Engine that is running the ACNS 5.2.1 software and later releases, and then enable RealMedia proxy caching and RealMedia transparent caching on the Content Engine. With the Setup utility, you can configure the Content Engine to accept redirected RTSP requests from a WCCP Version 2-enabled router. With the Content Engine CLI, you can configure the Content Engine to accept redirected RTSP requests from a WCCP Version 2-enabled router or a Layer 4 switch. RealProxy is configured with the RealSystem administrator GUI, which is accessed from the RealProxy window of the Content Engine GUI. (For information about access the RealSystem administrator GUI, see the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21.)
Chapter 8  Configuring RealMedia Services on Standalone Content Engines

Configuring RealMedia Services

Note

For information about how to use the Setup utility to enable RealMedia caching on a standalone Content Engine, see the “Configuring a Basic Configuration on Standalone Content Engines with the Setup Utility” section on page 4-10.

This section describes how to configure RealMedia streaming and caching services for a standalone Content Engine through the Content Engine CLI and the RealSystem Administrator GUI. The following topics are described in this section:

- Enabling RealProxy on Standalone Content Engines, page 8-13
- Configuring the RTSP Gateway for Standalone Content Engines, page 8-14
- Configuring RTSP Transparent Redirection and Caching of RealMedia Requests, page 8-17
- Configuring Direct Proxy Routing and RealMedia Proxy Caching, page 8-21
- Configuring RealProxy with the RealSystem Administrator GUI, page 8-21

When configuring RealMedia streaming and caching services with standalone Content Engines, note the following important points:

- In order to support RealMedia transparent caching, WCCP Version 2 must be running on the standalone Content Engine. WCCP Version 2 must also be running on the routers that will be redirecting RTSP requests to the Content Engine.
- You must configure disk space to include mediafs storage with the disk config command before you can run cache streaming media using RealProxy. The mediafs partitions is mounted on the standalone Content Engine. This is the storage partition that is used to store any RTSP streaming media content that is cached on the Content Engine.
- You have the IP address of the WCCP Version 2-enabled routers if you want to use transparent WCCP redirection.
- You have the IP address of the standalone Content Engine that you want to enable and run RealProxy on.
- You have a RealProxy license key. For information about the RealProxy license, see the “About Live Splitting and Caching VOD Files with RealProxy” section on page 8-6.
- Live broadcasts are live streams and not files, and therefore cannot be cached. However, RealProxy can split live broadcast (live splitting) to conserve network bandwidth. For more information about live splitting, see the “About Live Splitting and Caching VOD Files with RealProxy” section on page 8-6.

Note

Content Engines that use a Content Service Switch (CSS) to load balance streaming traffic cannot stream UDP traffic (such as RTSPU), because the Content Service Switch does not support UDP traffic.

Figure 8-2 provides a detailed view of how to configure RealMedia streaming and caching services initially for standalone Content Engines.
Figure 8-2  Configuration of RealMedia Streaming and Caching Services

Begin configuration of RealMedia services for a standalone Content Engine

Use the Setup utility or Content Engine CLI to enable RealProxy on the standalone Content Engine
- Accept the RealProxy license agreement
- Specify a RealProxy license key (accept the evaluation RealProxy license key, or enter your permanent key)
- Turn on the RealProxy feature on this Content Engine

Use the Content Engine CLI to specify the following settings for the RTSP gateway (the single point of entry for RTSP messages) that runs on the standalone Content Engine and is automatically enabled
- Specify the IP address of the RTSP gateway if the Content Engine is behind a NAT-enabled router (required)
- Change the default basic settings, such as the RTSP incoming port number (optional)
- Change the default advanced settings, such as the maximum request rate (optional)

Configure one or more of the following routing methods to direct RealMedia client requests to the RTSP gateway on the standalone Content Engine

- WCCP routing or Layer 4 switching (RTSP transparent redirection of RealMedia requests)
- Direct proxy routing (nontransparent)

WCCP

Configure RTSP redirection through WCCP (service 80)
- Configure on WCCP routers that will support this media service
- Configure on this Content Engine

Layer 4 switching

Configure RTSP redirection through Layer 4 switching
- Configure on Layer 4 switch
- Configure on this Content Engine

RealMedia transparent caching (caching VOD files) and live splitting with RealProxy

Manually configure RealMedia players to point directly to this Content Engine (nontransparent forward proxy server)

RealMedia proxy caching (caching VOD files) and live splitting with RealProxy

Use the RealSystem Administrator GUI to configure RealProxy on this Content Engine (for example, configure live splitting)

Configuration of RealMedia services for a standalone Content Engine is complete
Table 8-4 is a checklist of tasks for configuring RealMedia services for standalone Content Engines that are running the ACNS 5.2.1 software and later releases. This checklist includes the steps involved in configuring these services on a standalone Content Engine, including the configuration of one or more routing methods to direct RTSP requests from RealMedia players to the Content Engine.

**Table 8-4  Checklist for Configuring RealMedia Streaming and Caching Services for Standalone Content Engines**

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
</table>
| 1. Enable the RealProxy feature on the standalone Content Engine.  
   a. Accept the RealProxy license agreement.  
   b. Accept the evaluation RealProxy license, or specify your Cisco permanent RealProxy license.  
   c. Enable the licensed RealProxy feature on the standalone Content Engine. | See the “Enabling RealProxy on Standalone Content Engines” section on page 8-13. |
| 2. If necessary, specify the RTSP gateway settings.  
   a. If the Content Engine is behind a NAT-enabled router, you must specify the IP address of the RTSP gateway (required).  
   b. You can also change the default basic and advanced RTSP gateway settings (optional). | See the “Configuring the RTSP Gateway for Standalone Content Engines” section on page 8-14. |
| 3. Configure one or more of the following routing methods to direct content requests from RealMedia players to the RTSP gateway on this standalone Content Engine:  
   – Direct proxy routing (nontransparent)  
   – RealMedia RTSP transparent redirection (WCCP Version 2 routing or Layer 4 switching) | With direct proxy routing, the RealMedia client players (for example, RealPlayer or RealOne player) send their content requests directly to this Content Engine (acting as a nontransparent forward proxy server). With direct proxy routing, you must point the RealMedia clients directly to the Content Engine. See the “Configuring Direct Proxy Routing and RealMedia Proxy Caching” section on page 8-21.  
   With RealMedia RTSP transparent redirection, you must configure the WCCP routers or Layer 4 switches and the Content Engine (transparent proxy server) for this type of transparent redirection. See the “Configuring RTSP Transparent Redirection and Caching of RealMedia Requests” section on page 8-17. |
| 4. Use the RealSystem administrator GUI to configure RealProxy (for example, configure live splitting with RealProxy). | See the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21. |

**Tip**  
Live broadcasts are live streams and are not files; therefore, they cannot be cached. However, RealProxy can split live broadcasts (live splitting) to conserve network bandwidth.
Enabling RealProxy on Standalone Content Engines

The RealNetworks, Inc. RealProxy product is licensed software. To activate the licensed RealProxy feature on a standalone Content Engine, you must have a RealProxy license key. For more information about the RealProxy license, see the “About the RealProxy License Key” section on page 8-9.

Before enabling licenses for streaming media services on a Content Engine, make sure that your Content Engine clock and calendar settings are correct; otherwise, you will see an error message and the services will fail to install. Use the `show clock` EXEC command to display the system clock. To set the system clock, use the `clock set` EXEC command.

To use the Content Engine CLI to enable the licensed RealProxy feature on a standalone Content Engine, follow these steps:

**Step 1** View the RealProxy license agreement.
```
ContentEngine# show rtsp proxy media-real license-agreement
```

**Step 2** After reading the license agreement, enter global configuration mode and accept the license agreement.
```
ContentEngine# configure terminal
ContentEngine(config)# rtsp proxy media-real accept-license-agreement
```

**Step 3** Enter your Cisco license key for the licensed RealProxy feature.
```
ContentEngine(config)# rtsp proxy media-real license-key licensekey
```
Alternatively, accept an evaluation RealProxy license.
```
ContentEngine(config)# rtsp proxy media-real evaluate
```

**Step 4** Enable the licensed RealProxy feature on this Content Engine.
```
ContentEngine(config)# rtsp proxy media-real enable
```

---

**Note**
For information about uninstalling the RealProxy license on standalone Content Engines, see the “Uninstalling the RealProxy License Key” section on page 8-31.

The next step is to specify the RTSP gateway settings, if necessary. Because the RTSP gateway is automatically enabled on the Content Engine with a default configuration (see Table 8-5), you only need to change the default RTSP gateway settings in the following situations:

- If the Content Engine is behind a NAT-enabled router, you must specify the IP address of the RTSP gateway. By default, there is no IP address specified for the RTSP gateway.
- If you want to change any of the default settings, including the port that the RTSP gateway is to listen on for incoming requests (port 554 is the default).

For information about how to change the default RTSP gateway settings, see the “Configuring the RTSP Gateway for Standalone Content Engines” section on page 8-14.
Otherwise, the next step is to configure one or more of the following routing methods to direct content requests from RealMedia players to this standalone Content Engine:

- Direct proxy routing (nontransparent)
  
  With direct proxy routing, the RealMedia players send their requests directly to this Content Engine (nontransparent forward proxy server). For instructions on how to configure a RealMedia player on the end user desktops to point directly to this Content Engine as its proxy server, see the “Pointing RealMedia Players Directly to a Standalone Content Engine” section on page 4-46.

- WCCP routing or Layer 4 switch (RealMedia RTSP transparent redirection)

  By default, Layer 4 switching is not enabled and WCCP transparent redirection is not configured on the RTSP gateway. (See Table 8-5.) To enable transparent redirection of RealMedia RTSP requests through Layer 4 switching or WCCP Version 2, complete the process described in the “Configuring RTSP Transparent Redirection and Caching of RealMedia Requests” section on page 8-17.

### Configuring the RTSP Gateway for Standalone Content Engines

The RTSP gateway is a process that runs on the Content Engine. The RTSP gateway accepts an RTSP request and performs the initial RTSP handshake with RTSP-based clients (for example, RealMedia clients and Windows Media 9 players) on behalf of the backend RTSP servers (for example, the RealProxy server and the WMT RTSP server) that are running on the Content Engine.

**Note**

On standalone Content Engines that are running the ACNS 5.2 software and earlier releases, RealProxy is the only backend RTSP server that can be enabled on a standalone Content Engine.

In the ACNS 5.3.1 software release, the RTSP gateway was expanded to enable it to switch and tunnel RTSP requests from Windows Media 9 players to a WMT RTSP-based server. Consequently, you can enable RealProxy or the WMT RTSP-based server as a backend RTSP server on standalone Content Engines that are running the ACNS 5.3.1 software and later releases. For Content Engines that are registered with a Content Distribution Manager, you can also enable RealSubscriber and Cisco Streaming Engine as backend RTSP servers that run on the Content Engine.

After successful completion of uniformity checks, the RTSP gateway tunnels the request to the appropriate backend RTSP server that is running on the Content Engine. The RTSP gateway can tunnel the request to RealProxy, RealSubscriber, or the Cisco Streaming Engine on the Content Engine, depending on the requested media type, the backend RTSP servers that is currently enabled on the Content Engine, and the media player that is requesting the content.

After the RTSP gateway tunnels the request to a particular backend RTSP server that is running on the Content Engine and the backend server and the client negotiate the UDP ports, the RTSP gateway continues with RTSP message passing (SETUP). When the RTSP client issues a PLAY request, the streaming server starts streaming the data to the client over UDP.
Based on the properties of the incoming request, including such properties as user agent, final destination, and media file type, the RTSP gateway performs the following tasks with standalone Content Engines:

- Forwards the incoming request to the appropriate backend RTSP server (the RealProxy server or the WMT RTSP server) that is running on the Content Engine:
  - Forwards requests to the RealProxy server if the client is a RealMedia player. (The Content Engine uses RealNetworks’ proprietary RTSP as the protocol to serve the content to the media player.)
  - Forwards requests to the WMT RTSP server if the client is a Windows Media 9 player. (The Content Engine uses the IETF standard RTSP protocol plus proprietary Microsoft extensions to server the content to Windows Media 9 players.)
- Redirects the incoming request.
- Rejects the incoming request.

If the Content Engine is registered with a Content Distribution Manager, the RTSP gateway also redirects the incoming requests to other backend RTSP servers (for example, RealSubscriber or Cisco Streaming Engine) that are configured on the Content Engine.

Network Address Translation (NAT) is designed for IP address simplification and conservation because it enables private IP internetworks that use nonregistered IP addresses to connect to the Internet. NAT operates on a router, usually connecting two networks together, and translates the private addresses in the internal network into legal addresses before packets are forwarded onto another network. As part of this functionality, NAT can be configured to advertise only one external address for the entire network. This provides additional security, effectively hiding the entire internal network behind that one external address. NAT has the dual functionality of security and address conservation, and is typically implemented in remote access environments.

**Note**

If the Content Engine is behind a NAT-enabled router, you must specify the IP address of the RTSP gateway that is running on the Content Engine. By default, no IP address is specified.

**Default RTSP Gateway Settings**

The RTSP gateway is automatically enabled on the Content Engine, and cannot be disabled with a command. **Table 8-5** lists the default settings for the RTSP gateway.

<table>
<thead>
<tr>
<th>RTSP Gateway Setting</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address of RTSP gateway</td>
<td>Not specified</td>
</tr>
<tr>
<td>Incoming RTSP port</td>
<td>Port 554</td>
</tr>
<tr>
<td>Incoming RTSP request rate</td>
<td>40 requests per second</td>
</tr>
<tr>
<td>Layer 4 switching</td>
<td>Not enabled</td>
</tr>
<tr>
<td>WCCP transparent interception</td>
<td>Not configured</td>
</tr>
<tr>
<td>Maximum initial setup delay</td>
<td>10 seconds</td>
</tr>
<tr>
<td>Maximum request rate</td>
<td>40 requests per second</td>
</tr>
</tbody>
</table>
Configuring Basic Settings for the RTSP Gateway

By default, the RTSP gateway is always enabled on a Content Engine, and cannot be disabled by entering a CLI command. As Table 8-5 shows, the RTSP gateway has a set of default settings. You only need to change these default settings under the following conditions:

- You want to configure the RTSP gateway to listen for incoming RTSP requests on a port other than the default port (port 554).
- The Content Engine is behind a NAT-enabled router. In this case, you must specify the IP address of the RTSP gateway. By default, an IP address for the RTSP gateway is not specified.

To configure the basic settings for the RTSP gateway on a standalone Content Engine, use the `rtsp` global configuration commands.

**Table 8-6 Parameters for the rtsp Command**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip-address</td>
<td>Configures the IP address of the RTSP gateway.</td>
</tr>
<tr>
<td>rtsp-gateway-ip-address</td>
<td>IP address of the RTSP gateway that is running on the Content Engine. By default, no IP address is specified.</td>
</tr>
<tr>
<td>L4-switch enable</td>
<td>Enables Layer 4 switch interoperability with RTSP.</td>
</tr>
<tr>
<td>port</td>
<td>Port number on which the RTSP gateway is to listen for incoming RTSP requests. You can specify a single incoming port number. The port number can be from 1 to 65535. The default is port 554.</td>
</tr>
</tbody>
</table>

To configure the basic RTSP gateway parameters on a standalone Content Engine, follow these steps:

**Step 1**
Specify the incoming port (the port on which the RTSP gateway is to listen for incoming RTSP requests).
```
ContentEngine(config)# rtsp port incoming port-number
```

**Step 2**
Specify the IP address of the RTSP gateway.
```
ContentEngine(config)# rtsp ip-address rtsp-gateway-ip-address
```

**Step 3**
Configure transparent redirection of RTSP requests through WCCP Version 2 or Layer 4 switching.
```
For more information, see the “Configuring RTSP Transparent Redirection and Caching of RealMedia Requests” section on page 8-17.
```
Configuring Advanced Options for the RTSP Gateway

If the Content Engine is running the ACNS 5.2.1 software and later releases, you can use the `rtsp advanced` global configuration command to configure the following three advanced options for the RTSP gateway:

- Bypass gateway
- Maximum initial setup delay
- Maximum request rate

The syntax for the command is as follows:

```
rtsp advanced { bypass-gateway media-real | max-initial-setup-delay time_delay | max-request-rate number }
```

Table 8-7 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>advanced</td>
<td>Configures the advanced options for the RTSP gateway that is running on the Content Engine.</td>
</tr>
<tr>
<td>bypass-gateway</td>
<td>Sets the bypass gateway feature that enables the specified types of RTSP requests (for example, RealMedia requests) to bypass the RTSP gateway.</td>
</tr>
<tr>
<td>media-real</td>
<td>However, if the Content Engine is registered with a Content Distribution Manager, you can specify the following additional options for Content Engines because the Cisco Streaming Engine and RealSubscriber are supported as RTSP-based backend servers: the <code>bypass-gateway cisco-streaming-engine</code> option and the <code>bypass-gateway real-subscriber</code> option.</td>
</tr>
<tr>
<td>max-initial-setup-delay</td>
<td>Sets the maximum delay that is allowed between the TCP accept and the first RTSP message from the client. The unit of measurement is seconds. The default is 10 seconds.</td>
</tr>
<tr>
<td>time_delay</td>
<td>Maximum time delay allowed in seconds (0-2147483647). The default value is 10 seconds.</td>
</tr>
<tr>
<td>max-request-rate</td>
<td>Sets the maximum number of incoming requests per second that the RTSP gateway allows.</td>
</tr>
<tr>
<td>number</td>
<td>Maximum number of incoming requests per second that the RTSP gateway allows. The default value is 40 requests per second.</td>
</tr>
</tbody>
</table>

Configuring RTSP Transparent Redirection and Caching of RealMedia Requests

If RTSP transparent redirection through WCCP Version 2 or Layer 4 switching is being used to direct content requests from RealMedia players to a standalone Content Engine, you can configure the Content Engine to support RealMedia transparent caching for VOD files. In this case, the standalone Content Engine is acting as a transparent proxy server for the end users who are using RealMedia players to request streaming media content. After receiving a transparently redirected RTSP request from a RealMedia player, the Content Engine retrieves the requested content from the origin streaming server if it is not already stored in its local cache, stores a copy locally whenever possible (VOD files can be cached but not live streams), and sends the requested content to the RealMedia player.
The term RealMedia RTSP transparent redirection is used to refer to RTSP transparent redirection of RealMedia requests (WCCP service 80). In contrast, the term WMT RTSP transparent redirection is used to refer to RTSP transparent redirection RTSP transparent redirection (WCCP services 80 and 83) of WMT requests from Windows Media 9 players. For information about how to configure WMT RTSP transparent redirection, see the “Configuring RTSP Transparent Redirection of WMT Requests” section on page 9-31.

The Content Engine listens for redirected RTSP requests on the standard RTSP port (default port 554). To intercept RealMedia RTSP traffic on multiple ports, you must configure a user-defined WCCP service (services 90 to 97). For information on this topic, see the “Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 6-15.

With RealMedia RTSP transparent redirection, a Layer 4 switch or WCCP Version 2-enabled router redirects RealMedia requests to the Content Engine (acting as a transparent proxy server). RTSP transparent redirection is used to support RealMedia transparent caching on a standalone Content Engine.

To enable transparent redirection of RTSP requests through Layer 4 switching, enter the rtsp L4-switch enable global configuration command. After you enter the command, a message appears indicating that Layer 4 switching for RTSP has been enabled on the Content Engine:

```
ContentEngine(config)# rtsp L4-switch enable
Turn on l4 switch
```

To configure RealMedia RTSP transparent redirection through WCCP Version 2, you must perform these tasks:

- Configure RealMedia RTSP redirection on the WCCP Version 2 routers that will support this RTSP streaming service.
- Configure RTSP redirection on the standalone Content Engine that will be receiving these transparently redirected RTSP requests.

The RealMedia RTSP redirection service (service 80) is a WCCP Version 2 standard media caching service that supports the transparent redirection of RTSP requests from RealMedia players. This WCCP service permits WCCP Version 2-enabled routers to redirect RTSP requests from RealMedia players transparently to a single port (port 554) on a Content Engine. After receiving a redirected RTSP request, the Content Engines checks if whether it has a cached copy of the requested content. If it has, the Content Engine sends the RealMedia player the requested content. Otherwise, the Content Engine retrieves the requested content from the origin streaming server, caches a copy locally if RealMedia transparent caching is enabled on the Content Engine, and sends the RealMedia player the requested content.

To configure RealMedia RTSP transparent redirection on multiple ports, you must configure a user-defined WCCP service (services 90 to 97) on the WCCP Version 2-enabled routers and the Content Engine. For more information about configuring such WCCP services, see the “Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 6-15.
The following example shows how to configure RealMedia RTSP transparent redirection (service 80) on a standalone Content Engine and a router with WCCP Version 2, and enable RealProxy transparent caching on the Content Engine. This example assumes that you have enabled the licensed RealProxy feature on the standalone Content Engine, as described in the “Enabling RealProxy on Standalone Content Engines” section on page 8-13.

To configure RealMedia RTSP transparent redirection (the rtsp service [service 80]) and RealMedia transparent caching for standalone Content Engines, follow these steps:

**Step 1** Enable WCCP Version 2 on the router. (WCCP Version 1 does not support service 80.)

```
Router# configure terminal
Router(config)# ip wccp version 2
```

**Step 2** Enable service 80 on the WCCP Version 2-enabled router.

```
Router(config)# ip wccp 80
```

**Step 3** Specify the interface on which the RTSP redirection service will run.

```
Router(config)# interface type number
```

The following shows how to configure the router to run service 80 on the Fast Ethernet interface:

```
Router(config)# interface fastEthernet 0/0
```

**Step 4** Configure the router to use the outbound interface for service 80.

```
Router(config-if)# ip wccp 80 redirect out
```

**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 (using the `ip wccp service number redirect in` interface configuration command) if the router supports the redirection in feature.

**Step 5** End the configuration session on the router.

```
Router(config-if)# end
```

**Step 6** Enable RTSP redirection through WCCP on the standalone Content Engine. This is the Content Engine that will act as the transparent proxy server for redirected RTSP requests from these WCCP Version 2-enabled routers.

a. Create the numbered router list that you want to associate with service 80.

   In the following example, there is a single WCCP Version 2-enabled router associated with router list 1. This router has an IP address of 10.1.3.1.

   ```
   ContentEngine(config)# wccp router-list 1 10.1.3.1
   ```

b. Enable the router list (router list 1) that you just created in the previous step (Step a.).

   ```
   ContentEngine(config)# wccp rtsp router-list-num 1
   WCCP configuration for RTSP succeeded. Please remember to configure WCCP service 80 on the corresponding router.
   ```

   You have already configured service 80 on the corresponding router (the router with an IP address of 10.1.1.1) in Step 2 through Step 5 of this procedure.
c. Enable WCCP Version 2 on the Content Engine that will accept redirected RTSP requests from the WCCP Version 2-enabled routers that are listed in router list 1.

```
ContentEngine(config)# wccp version 2
```

**Step 7** If transaction logging is not currently enabled on the Content Engine, enable it.

```
ContentEngine(config)# transaction-log enable
```

**Step 8** Save the new configuration on the Content Engine.

```
ContentEngine# copy running-config startup-config
```

**Step 9** Display the list of WCCP services that are currently configured on Content Engine A to verify that service 80 (the rtp service) is listed.

```
ContentEngineA# show wccp services
```

```
Services configured on this Content Engine
    Web Cache
    RTSP
    FTP
```

The partial sample output that is shown above indicates that this WCCP service is now enabled on the Content Engine, along with the FTP service and the web-cache service. For a descriptive list of the supported WCCP services, see Table B-3.

**Step 10** Verify that WCCP transparent redirection is now enabled on the Content Engine.

```
ContentEngine# show rtsp
```

**Step 11** If necessary, specify the RTSP gateway settings:

a. If the Content Engine is behind a NAT-enabled router, you must specify the IP address of the RTSP gateway (required).

b. You may also want to change the default basic and advanced RTSP gateway settings.

For more information, see the “Configuring the RTSP Gateway for Standalone Content Engines” section on page 8-14.

**Step 12** After configuring the routers and Content Engine to support RTSP redirection through WCCP Version 2, you can configure RealMedia transparent caching on the Content Engine.

a. Enable the RealProxy product feature on the Content Engine, if it is not already enabled.

```
ContentEngine(config)# rtsp proxy media-real enable
```

b. Use the RealSystem administrator GUI to configure RealProxy (for example, for live splitting). For more information, see the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21.

For an example of how to verify whether RealMedia transparent caching is working properly for standalone Content Engines, see the “Example 1—Verifying the Configuration for RealMedia VOD Caching” section on page 8-24.
Configuring Direct Proxy Routing and RealMedia Proxy Caching

If direct proxy routing is being used to direct content requests from RealMedia players directly to the standalone Content Engine, then you can configure the Content Engine to support RealMedia proxy caching for VOD files. With RealMedia proxy caching, the standalone Content Engine is functioning as a nontransparent proxy server for the RealMedia players. After receiving a content request directly from a RealMedia player (for example, a RealPlayer or RealOne player), the Content Engine retrieves the requested VOD file from the origin streaming server if it is not already stored in its local cache, stores a copy locally, and sends the requested streaming media content to the RealMedia player.

The following example assumes that you have enabled the licensed RealProxy feature on the standalone Content Engine, as described in the “Enabling RealProxy on Standalone Content Engines” section on page 8-13.

To use the Content Engine CLI to configure RealMedia proxy caching on a standalone Content Engine, follow these steps:

**Step 1** Configure the RealMedia players to send their requests directly to this Content Engine.

By default, the RTSP gateway on the Content Engine listens for incoming RTSP requests on port 554. If you entered the `rtsp port incoming port-number` global configuration command to specify another port as the incoming RTSP port (for instance, you configured the RTSP gateway to listen on port 575 instead of port 554), you must configure the RealMedia players to send their requests directly to the configured RTSP incoming port. For more information, see the “Pointing RealMedia Players Directly to a Standalone Content Engine” section on page 4-46.

*Note* If a firewall is positioned between a Content Engine and a requesting client, make sure that you specify the external IP address of the Content Engine as the proxy server when you configure the RealMedia proxy settings on client desktops. See the “Pointing RealMedia Players Directly to a Standalone Content Engine” section on page 4-46.

**Step 2** Use the RealSystem administrator GUI to configure RealProxy (for example, for live splitting) on the Content Engine.

For more information, see the next section, “Configuring RealProxy with the RealSystem Administrator GUI.”

**Configuring RealProxy with the RealSystem Administrator GUI**

RealProxy is a licensed product from RealNetworks, Inc. You use the Setup utility or the Content Engine CLI to enable the licensed RealProxy feature on a standalone Content Engine. You can also use the Setup utility or the Content Engine CLI to enable RealMedia proxy caching and RealMedia transparent caching on a standalone Content Engine after RealProxy has been enabled on the Content Engine. However, you perform RealProxy configuration through the RealNetworks RealSystem administrator GUI.
To access the RealSystem administrator GUI, follow these steps:

**Step 1** From the Content Engine GUI, choose **Caching > Real Proxy**. The Content Engine RealProxy window appears. (See Figure 8-3.)

**Note** To access the Content Engine GUI, enter the Content Engine IP address in secure mode and append the default port number 8003 as the URL address in your browser of choice. For example, enter `https://ContentEngineIPaddress:8003` as the URL.

**Figure 8-3  Content Engine RealProxy Window**

![Content Engine RealProxy Window](image)

**Note** The **ADMIN** button only appears in the Content Engine RealProxy window if the RealProxy software has been installed and enabled on the Content Engine.

**Step 2** In the Content Engine RealProxy window, click the **ADMIN** button.

**Step 3** Use admin as the default username and diamond as the password to access the RealSystem administration GUI from the Content Engine RealProxy window. The main window for the RealSystem administrator GUI (see Figure 8-4) appears.
Step 4 Use the RealSystem administrator GUI to configure the licensed RealProxy feature on this Content Engine. For example, use this GUI to configure RealProxy live splitting and caching of VOD files.

After configuring RealProxy, you should verify that the RealProxy configuration is working properly. For some examples of how to verify the RealProxy configurations for live splitting and VOD caching, see the next section, “Verifying RealProxy Configurations for Standalone Content Engines.”

Verifying RealProxy Configurations for Standalone Content Engines

This section provides two examples of how to verify the configuration of RealProxy on standalone Content Engines:

- Example 1—Verifying the Configuration for RealMedia VOD Caching, page 8-24
- Example 2—Verifying the Configuration for RealProxy Live Splitting, page 8-27
Example 1—Verifying the Configuration for RealMedia VOD Caching

This first example shows how to verify the configuration of the RealMedia transparent caching service on a standalone Content Engine. This example assumes the following:

- RealProxy has been enabled on the Content Engine. (See the “Enabling RealProxy on Standalone Content Engines” section on page 8-13.)
- There is a single standalone Content Engine (Content Engine A) that has WCCP Version 2 enabled on it.
- There is a single WCCP Version 2-enabled router (Router A) that is configured to redirect content requests from RealMedia players to Content Engine A (transparent proxy server).
- The RealMedia transparent caching service is enabled on Content Engine A, and Content Engine A is configured to accept redirected RTSP requests from Router A.
- The RealMedia players (in this case, RealPlayer) on Client A and Client B are not configured to point directly to Content Engine A.
- Client A and Client B are on the same subnet.

To verify that the RealMedia transparent caching service (RealMedia transparent caching through WCCP Version 2) is working properly, follow these steps:

Step 1  From the Client A desktop, use RealPlayer to request a RealMedia streaming video file (.rm file).
  a. From the RealPlayer menu, choose File > Open URL.
  b. Specify a URL that points to a RealMedia streaming video file (for example, rtsp://origin-streaming-server-ip-address/gm1_real_02_00500.rm).

  Note  Request the video file more than once. The origin streaming server from which you are requesting content (for example, the .rm video files) must be on a different subnet than Client A and Client B, so that the RealPlayer requests from these client desktops are routed to Router A.

  The requested video file should start playing in RealPlayer on Client A.

Step 2  Check the statistics on RealPlayer.
  a. From RealPlayer, choose Tools > Playback Statistics.
  b. In RealPlayer, click the Streams tab to bring it to the front.
  c. In the Streams tab, check whether UDP is shown as the transport protocol that is being used. If UDP is shown as the transport protocol, this indicates that the stream is being delivered to RealPlayer in a streaming fashion instead of reverting to HTTP when the streaming proxy or server is not available.

Step 3  From the Client A desktop, use RealPlayer to replay the same streaming video file that you just requested.
Step 4  On the WCCP-enabled router, check the number of redirected RTSP packets from RealMedia players. The following is a sample output:

Router# show ip wccp 80
Global WCCP information:
  Router information:
    Router Identifier:           10.1.3.1
    Protocol Version:            2.0
  Service Identifier: 80
    Number of Cache Engines:     1
    Number of routers:           1
    Total Packets Redirected:    6
    Redirect access-list:        -none-
    Total Packets Denied Redirect: 0
    Total Packets Unassigned:   0
    Group access-list:          -none-
    Total Messages Denied to Group: 0
    Total Authentication failures: 0

Step 5  On the Content Engine, check the RealMedia caching statistics.

a. From Content Engine A, display the RealMedia caching saving statistics for this Content Engine.

ContentEngine# show statistics rtsp proxy media-real savings
  Media Cache Statistics - Savings
  Requests                      Bytes
  ----------------------------------------------
  Total:                      17               16666028
  Hits:                       11               3656524
  Miss:                       6                13009504
  Savings:                    64.7 %             21.9 %

b. From Content Engine A, display the number of RealMedia requests that this Content Engine has received. (See the sample output.) Verify that the Content Engine is processing RealMedia requests.

ContentEngine# show statistics rtsp proxy media-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

Step 6  From the Client B desktop, use RealPlayer to request the same video file that you requested earlier from the Client A desktop.

This will allow you to check whether Content Engine A is storing a copy of the requested VOD in its local cache instead of retrieving the video file again from the origin streaming server.

- The number of cache hits displayed in the output of the show statistics rtsp proxy media-real savings EXEC command should increase as you use RealPlayer on Client B to request the same video file that you just requested from RealPlayer on Client A.

- The number of RealMedia requests displayed in the output of the show statistics rtsp proxy media-real requests EXEC command should increase as you use RealPlayer on Client B to request the same VOD file that you just requested from Client A.
Chapter 8  Configuring RealMedia Services on Standalone Content Engines

Verifying RealProxy Configurations for Standalone Content Engines

Step 7  On Router A, open a console or Telnet session.

Step 8  On Router A, display statistics and status information for the rtsp service (service 80).

```
Router# show ip wccp 80
```

The statistics should show a number greater than 0 for packets redirected.

Step 9  Verify that packets are being redirected to Content Engine A from Router A:

- If Router A shows that there are packets being redirected to Content Engine A, then RealMedia transparent caching (transparent redirection of RTSP requests through service 80) is operating properly on Content Engine A and Router A.
- If Router A shows that no packets are being redirected to Content Engine A, then RealMedia transparent caching is not operating properly. In this case, you should troubleshoot the problems with your configuration of the rtsp service. The following are some examples of how to do this.

  a. From Content Engine A, display the list of WCCP services that are currently configured on Content Engine A. The following is a sample output.

```
ContentEngineA# show wccp services
Services configured on this Content Engine
  Web Cache
  RTSP
  FTP
```

Verify that the rtsp service (service 80) is listed.

b. From Content Engine A, display a list of WCCP-enabled routers that recognize Content Engine A. The following is a sample output:

```
ContentEngineA# show wccp routers
Routers Seeing this Content Engine
  Router Id      Sent To
     10.0.0.0      10.1.1.1
Routers not Seeing this Cache Engine
     10.1.1.1
Routers Notified of but not Configured
-NONE-
```

Check the command output to determine if Router A is on the list of WCCP-enabled routers that recognizes Content Engine A.

c. From Content Engine A, display WCCP generic routing encapsulation (GRE) packet-related information for Content Engine A.

```
ContentEngineA# show wccp gre
```

Check the command output to view the number of redirected packets that Content Engine A has rejected and accepted. Verify that the number of accepted packets is increasing as you continue to request VOD files that are on streaming servers that are on different subnets than the requesting client (RealPlayer on the Client A and B desktops).
Step 10  You can also use the RealSystem administrator GUI to monitor RealProxy statistics when RealMedia clients are connected.

   a. From the Content Engine GUI, choose **Caching > Real Proxy**. The Content Engine RealProxy window appears. (See Figure 8-3.)

   b. Click the **Admin** button.

   c. When asked, enter the username and password. The default username is admin. The default password is diamond. After entering a valid username and password, the RealSystem Administrator main window appears. (See Figure 8-4.)

   d. Use the RealSystem Administrator GUI to monitor RealProxy statistics for the RealMedia requests that are being serviced by RealProxy on this Content Engine.

Step 11  By default, transaction logging is enabled on the Content Engine. Verify that the RealMedia transactions that are being serviced by this Content Engine are being tracked in the RealProxy transaction log.

```
ContentEngine# type-tail /local1/logs/real-proxy/rproxy-transaction-log-filename
```

Depending upon where the sysfs is mounted, RealProxy logs are logged to a working log on the local disk in one of these files:

- /local1/logs/real-proxy/working.log
- /local2/logs/real-proxy/working.log

You can specify the interval at which the working log should be cleared by moving the data to an archive log. The archive log files are located on the local disk in the /local1/logs/ or /local2/logs/ directory depending upon where the sysfs is mounted.

Because multiple archive files are saved, the filename includes the time stamp when the file was archived. Because the files can be exported to an FTP/SFTP server, the filename also contains the IP address of this Content Engine. For more information about using transaction logs, see the “Using ACNS Software Transaction Logs” section on page 21-31.

Step 12  Check the system log file for RealProxy error messages.

```
ContentEngine# type-tail syslog.txt
```

RealProxy generates error messages and writes them to the RealProxy log file. These error messages are captured by the ACNS software and passed to the system log file. See Table 21-9 for correspondence between the RealProxy error codes and corresponding syslog priority levels. For more information about system logging, see the “Monitoring the Performance of Specific URLs” section on page 21-52.

Example 2—Verifying the Configuration for RealProxy Live Splitting

This second example shows how to verify the configuration for RealProxy live splitting (live splitting through RealProxy and a WCCP Version 2-enabled router) on a standalone Content Engine. When RealProxy is enabled on a standalone Content Engine, the Content Engine serves as the stream splitting point for all local users (RealMedia players) whose RealMedia traffic is directed to the Content Engine. All further requests to that origin streaming server are served by the Content Engine, which splits the stream and serves it to the RealMedia players. Live broadcasts (live streams) are not files and therefore cannot be cached. For more background information about RealProxy live splitting, see the “About Live Splitting and Caching VOD Files with RealProxy” section on page 8-6.
In this example, the RealMedia players on the Client A, B, and C desktops are configured to point to a live stream that is set up to play continuously. WCCP Version 2 is used to redirect the request for this live stream to the Content Engine. A WCCP Version 2-enabled router transparently intercepts the request for the live stream, and redirects the request to the Content Engine (Content Engine A) that is running RealProxy.

This example assumes the following:

- RealProxy is enabled on Content Engine A. (See the “Enabling RealProxy on Standalone Content Engines” section on page 8-13.)
- WCCP Version 2 is also enabled on Content Engine A.
- There is a single WCCP Version 2-enabled router (Router A) that is configured to redirect content requests from RealMedia players to Content Engine A (transparent proxy server).
- The RealMedia players (in this case, RealPlayer) on Clients A, B, and C are not configured to point directly to Content Engine A.
- Clients A, B, and C are on the same subnet.

To verify that RealProxy live splitting is working properly, follow these steps:

**Step 1** Create a live stream on an origin streaming server (a Helix Universal Server) that supports the RealNetworks proprietary RTSP protocol.

Use either an encoder or the Simulated Live Transfer Agent (SLTA) tool.

SLTA is a Helix Universal Server utility to stream a prerecorded clip as if it were a live event.

The origin streaming server (Helix Universal Server) on which you are creating the live stream should be on a different subnet from that of Clients A, B, and C so that the RealMedia requests from these clients are routed to Router A.

**Step 2** From the Client A desktop, use RealPlayer to request the live stream (live event).

a. From the RealPlayer menu, choose **File > Open URL**.

b. Specify a live event alias (for example, `rtsp://origin-streaming-server-ip-address/broadcast/live`).

The requested live event should start playing in RealPlayer on Client A.

**Step 3** From the Client B and Client C desktops, use RealPlayer to request the same live stream that you requested earlier with RealPlayer on the Client A desktop.

**Step 4** Use the same verification process that is described in the “Example 1—Verifying the Configuration for RealMedia VOD Caching” section on page 8-24 to verify the following:

- The WCCP-enabled router (Router A) is redirecting the RealPlayer requests for live streams to the Content Engine A.
- Content Engine A is serving the live streams with bandwidth savings to Client A and Client B.
- Transactions related to these live streams (for example, which clients are viewing this particular live event, and the length of time that the client viewed this live event) are being logged in the RealProxy transaction log on Content Engine A.
For example, use this process:

a. Open a console or Telnet session on Router A. On Router A, enter the `show ip wccp 80` EXEC command to display statistics and status information for service 80 for Router A. The statistics should show a number greater than 0 for packets redirected.

b. From Content Engine A, enter the `show statistics rtsp proxy media-real requests` EXEC command to display the number of RealMedia requests that Content Engine A has received. The statistic should show a number greater than 0 for live split requests.

---

**Restoring the RealProxy Factory-Default Configuration on Standalone Content Engines**

On standalone Content Engines, RealProxy is enabled through the Content Engine CLI or through the Setup utility. To change the RealProxy default configuration, you must use the RealNetworks RealSystem administrator GUI, which you can access from the Content Engine GUI. However, in the ACNS 5.3.3 software and later releases, you can use the Content Engine CLI (the `rtsp real-proxy restore factory-default` EXEC command) to restore the RealProxy factory-default configuration. The default configuration is the RealProxy configuration file and the database that contains the RealProxy license keys on a standalone Content Engine.

In the ACNS 5.3.3 software release, the `rtsp real-proxy default-configuration` EXEC command was replaced with the `rtsp real-proxy restore factory-default` EXEC commands. In the ACNS 5.3.1 software and earlier releases, when you entered the `rtsp real-proxy default-configuration` EXEC command, only the RealProxy configuration files were restored to the default settings; the database that contains the RealProxy license key settings was not restored to the factory defaults.

---

**Note**

On Content Engines that are registered with a Content Distribution Manager, you can use the Content Distribution Manager or the CLI to restore the RealNetworks (RealProxy and RealSubscriber) license key settings to the factory default. For information about how to restore the factory-default settings for RealNetworks license keys on Content Engines that are registered with a Content Distribution Manager, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*.

To restore the RealProxy factory-default configuration on a standalone Content Engine, follow these steps:

**Step 1**

Enter the `rtsp real-proxy restore factory-default` EXEC command, and enter `yes` when asked if you want to proceed:

```
ContentEngine# rtsp real-proxy restore factory-default
User would lose the current real proxy configuration. Do you want to proceed? [yes/no] yes
Restart Real Proxy to load the factory defaults configuration.
```
Restarting RealProxy on Standalone Content Engines

To restart RealProxy on a standalone Content Engine, follow these steps:

**Step 1** Stop RealProxy on the Content Engine.

```
ContentEngine(config)# no rtsp proxy media-real enable
```

**Step 2** Restart RealProxy on the Content Engine.

```
ContentEngine(config)# rtsp proxy media-real enable
```

Disabling RealMedia Caching on Standalone Content Engines

To disable RealMedia caching on a standalone Content Engine, follow these steps:

**Step 1** Access the RealSystem Administrator GUI window by clicking the Admin button in the RealProxy window of the Content Engine GUI. (See Figure 8-3.) (You must enable the RealProxy before you can access the Admin button in this window.)

**Note** The administrator, usernames, and all associated passwords configured on the Content Engine are not the same as the ones contained in the RealProxy authentication database. Therefore, not all Content Engine users have access to the RealSystem Administrator GUI.

**Step 2** Choose Configure > Cache.

**Step 3** In the Enable Caching field, choose No.

**Step 4** Click Apply.
Step 5  Stop RealProxy on the Content Engine.

   ContentEngine(config)# no rtsp proxy media-real enable

Step 6  Restart RealProxy on the Content Engine.

   ContentEngine(config)# rtsp proxy media-real enable

---

**Uninstalling the RealProxy License Key**

If the RealProxy license key is no longer needed on the Content Engine because the licensed RealProxy feature is not needed, you can uninstall the RealProxy license key by entering the `no rtsp rproxy media-real license-key` global configuration command. After a license key is uninstalled on one Content Engine, it can be used on another Content Engine if that Content Engine supports the licensed RealProxy feature.

**Note**

The licensed RealProxy feature must be disabled using the `no rtsp proxy media-real enable` command before uninstalling the RealProxy license key on a standalone Content Engine.

---

**Displaying RealProxy Statistics for Standalone Content Engines**

You can use the `show statistics rtsp proxy media-real` EXEC commands to display RealProxy statistics for standalone Content Engines. The displayed statistics relate only to objects transported over RTSP that were requested by a RealMedia client. Objects transported over HTTP are counted in the HTTP statistics. Streaming objects requested by other clients or transported over protocols bypass the Content Engine.

ContentEngine# show statistics rtsp proxy media-real requests

```
Media Cache Statistics - Requests
-----------------------------------------------
Total Received Requests:          0          -
  Demand Cache Hit:            0          0.0
  Demand Cache Miss:          0          0.0
  Demand Pass-Through:       0          0.0
  Live Split:               0          0.0
  Live Pass-Through:        0          0.0
```

ContentEngine# show statistics rtsp proxy media-real savings

```
Media Cache Statistics - Savings
-----------------------------------------------
Total:          0          0
  Hits:          0          0
  Miss:          0          0
  Savings:      0.0 %      0.0 %
```

You can also obtain detailed configuration, statistics, and reporting of RealProxy status through the RealNetworks RealSystems administrator GUI. The Content Engine GUI has a RealProxy window (Figure 8-3). The ADMIN button is active in the Content Engine Management GUI when RealProxy is
installed and enabled on the Content Engine. You will be provided with a default username and password to access this administrator window from the Content Engine GUI. For more information, see the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21.
CHAPTER 9

Configuring WMT Streaming Media Services on Standalone Content Engines

This chapter provides an overview of the Windows Media Technologies (WMT) streaming and caching services, and describes how to use the Content Engine CLI to configure these services on standalone Content Engines.

This chapter contains the following sections:

- Overview of the Windows Media Services Streaming Solution, page 9-2
- Overview of the WMT Streaming and Caching Services, page 9-4
- Configuration Guidelines, page 9-10
- Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines, page 9-14
- Enabling WMT Licenses on Standalone Content Engines, page 9-17
- Configuring General WMT Settings on Standalone Content Engines, page 9-18
- Configuring Transparent Redirection of WMT Requests, page 9-31
- Enabling and Configuring WMT Caching on Standalone Content Engines, page 9-34
- Configuring Standalone Content Engines to Distribute VOD Files, page 9-35
- Configuring Standalone Content Engines to Deliver WMT Live Streams, page 9-37
- Displaying Information about the Current WMT Configuration, page 9-46
- Displaying Information about the WMT RTSP Server Configuration, page 9-46
- Using WMT Logging with Standalone Content Engines, page 9-47

Note

Throughout this chapter the following terminology is used. The Windows Media streaming and caching services are collectively referred to as the WMT feature. Windows Media Player 9 Series clients are called Windows Media 9 players. Windows Media Services 9 servers are called Windows Media 9 servers. The Windows Media Services 9 RTSP backend server that is running on the Content Engine is called the WMT RTSP server. The term WMT RTSP transparent redirection is used to refer to RTSP transparent redirection (WCCP Version 2 services 80 and 83) of WMT RTSP requests from Windows Media 9 players.
Overview of the Windows Media Services Streaming Solution

The Windows Media Services (WMS) is the Microsoft streaming solution for creating, distributing, and playing back digital media files on the Internet. Windows Media Services 9 Series (WMS 9) is the new Windows Media solutions from Microsoft.

Table 9-1 describes the major components of Windows Media Services.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Media player</td>
<td>Desktop application that the end user runs to play requested digital media files (for example, Windows Media 6.4 players and Windows Media 7.0 players, Windows Media 9 players or Windows Media 10 players). These clients can take advantage of the VCR-like controls in the Windows Media player to pause the stream or to skip backward or forward (in the case of stored content [video on demand]).</td>
</tr>
<tr>
<td>Windows Right Manager and Encoder</td>
<td>Content-creation application.</td>
</tr>
<tr>
<td>Windows Media Server</td>
<td>Server and distribution application that uses an application-level protocol called Microsoft Media Server (for example, Windows Media 9 server and Windows Media 4.1 server) to send active streaming format (ASF) files across the Internet.</td>
</tr>
</tbody>
</table>

With WMS 9, Microsoft introduced a major change in the streaming protocol. Windows Media Services 9 Series by default uses a new RTSP-based protocol for streaming.

These are the streaming protocols currently used by Windows Media 9 players:

- Windows Media Services 9 Series RTSP/RTP-based protocol
- Windows Media Services 9 Series-over-HTTP

As Figure 9-1 shows, a Content Engine, which is running the ACNS 5.5.1 software, has full interoperability with the Windows Media 9 server and a Windows Media 9 player over all of these streaming protocols.

Note: Although the Windows Media Player will play a file using the HTTP protocol (from port 80), the Content Engine does not support HTTP streaming because the Content Engine does not buffer HTTP files.
In the ACNS 5.3.1 software and later releases, RTSP/RTP is a supported streaming protocol. Consequently, RTSP requests from Windows Media 9 players are supported. Proxy caching (caching VOD files) and live splitting for WMT RTSP is supported. In centrally managed deployments (that is, Content Engines are registered with a Content Distribution Manager), managed live events are also supported. You use the Content Distribution Manager GUI to configure a managed live event. Standalone Content Engines do not support managed live events. End-to-end RTSP (from the client to the encoder) is not currently supported for managed live events. For information about configuring managed live events, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

When using direct proxy routing or WCCP redirection to route requests to standalone Content Engines, the unicast published URL can be the following:

- `rtsp://liveChannelOriginFqdn/program-name` (added in the ACNS 5.3.1 software release)

In the ACNS 5.3.1 software and later releases, the live stream source of a Windows Media live program can be one of the following

- `http://encoder:port-number`
- `rtsp://wmStreamingServer:port-number/file name` (RTSP support was added in the ACNS 5.3.1 software release)

For more information about WMT RTSP, see the “About the WMT RTSP Protocol” section on page 9-4.

The ACNS 5.2.1 software and later releases interoperate with the following software:

  
  In the ACNS 5.3.1 software to ACNS 5.4 software releases, WMT RTSP and WMT MMS are supported. In ACNS 5.5.x release, WMT MMS is not supported but WMT RTSP is supported.

- Windows Media Services 4.1 Series—Includes the Windows Media Player 4.1 Series, Windows Media Encoder, and Windows Media 4.1 server.

Content Engines that use a Content Service Switch (CSS) to load balance streaming traffic cannot stream UDP traffic (such as RTSPU), because the Content Service Switch does not support UDP traffic.
Overview of the WMT Streaming and Caching Services

When the WMT feature is enabled on the Content Engine, the Content Engine provides a native (integrated) WMT server that delivers Microsoft’s standard streaming formats (.ASF, .WMA, and .WMV files) through either unicast or multicast streams. The integrated WMT server has the ability to serve the streams to the clients by VOD, broadcast (live), and multicast. The WMT feature also allows a standalone Content Engine to support WMT transparent caching and WMT proxy caching.

The WMT feature on a standalone Content Engine is licensed software. To enable this feature on a Content Engine, you must have a WMT license key. You must specify a permanent license key that is supplied on a certificate shipped with the Content Engine, or use an evaluation key for a temporary period. If you are downloading the ACNS 5.x software, you can purchase a WMT license though the Cisco.com website. You specify the WMT license key as part of enabling the WMT feature on a standalone Content Engine. See the “Enabling WMT Licenses on Standalone Content Engines” section on page 9-17.

About the WMT RTSP Protocol

The Real-Time Streaming Protocol (RTSP) is a standard Internet streaming control protocol (RFC 2326). It is an application level protocol for control over the delivery of data with real-time properties such as video and audio. RTSP has been widely adopted in the industry. For example, Apple Computer’s QuickTime, RealNetworks’ RealMedia, and the Cisco Streaming Engine all use RTSP as the streaming control protocol. In WMS 9, Microsoft added support for the RTSP protocol as the streaming control protocol. In earlier versions of WMS (for example, WMS 4.1), WMS used MMS as the streaming control protocol. In the ACNS 5.3.1 software and later releases, WMT RTSP support for WMS 9 is available (that is, support for RTSP requests from Windows Media 9 players).

Note

In ACNS 5.5 software release, MMS is not supported. However, MMS-over-HTTP is supported.

The WMT RTSP server, which is running on the Content Engine, uses the WMT RTSP protocol to serve the VOD request to the Windows Media 9 players. The WMT RTSP protocol is the IETF RTSP standard protocol plus Microsoft proprietary extensions. The WMT RTSP server also uses this protocol to support broadcasting. The standard listening port for RTSP services is port 554.

RTSP requests from Windows Media 9 players can be routed directly to the Content Engine or transparently redirected. To route such requests directly to the Content Engine, you must configure the Windows Media 9 players to point directly to the Content Engine. To transparently redirect such requests to the Content Engine, you must configure WMT RTSP transparent redirection (WCCP Version 2 services 80 and 83) on the Content Engine and the WCCP Version 2 router. For more information about configuring Windows Media 9 players to point directly to a Content Engine, see the “Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests” section on page 4-43. For more information about configuring WMT RTSP transparent redirection, see the “Configuring RTSP Transparent Redirection of WMT Requests” section on page 9-31.

Tip

For live streaming, Content Engines always obtain the live stream from an external WMT server; the Content Engine is never the originator of the live content. For a standalone Content Engine to deliver WMT live streams, you need WMT caching proxy and server capabilities on the standalone Content Engine. The WMT product is licensed software and requires a WMT license key. For more information about this license key, see the “Enabling WMT Licenses on Standalone Content Engines” section on page 9-17.
How Standalone Content Engines Process WMT Requests

Standalone Content Engines can receive WMT requests directly from WMT clients, or from WCCP Version 2 routers or Layer 4 CSS switches (through WMT transparent redirection).

The actual protocol used is negotiated between the WMT client and the server. If both the client and the server are Windows Media Services 9 Series, then the RTSP protocol is used if the URL starts with mms://, and the HTTP protocol is used if the URL starts with http://. If either the client or the server is pre-WMS 9 (the client is a Windows Media 6.4 or 7.0 player, or the server is a Windows Media 4.1 server instead of a Windows Media 9 server), the MMS protocol is used.

In the case of MMS-over-HTTP with Windows Media Services 9 Series, a standalone Content Engine that is running the ACNS 5.3.1 software and later releases, supports the Fast Start and Fast Cache features for preloaded VOD, live-split, and on-demand (cache-hit) content from Windows Media 9 players over the following protocols: HTTP, RTSP, and MMS-over-HTTP. For more information on these features, see the “Configuring Standalone Content Engines to Deliver WMT Live Streams” section on page 9-37 and the “Configuring Fast Cache on Standalone Content Engines” section on page 9-29.

Note

Support for the Fast Start and Fast Cache features was added in the ACNS 5.2.1 software. In the ACNS 5.2.x software, these features are only available in MMS-over-HTTP streaming with Windows Media Services 9 Series. In the ACNS 5.3.1 software and later releases, the Fast Start and Fast Cache features are also available for RTSP requests from Windows Media 9 players (WMT RTSP requests).

In the ACNS 5.2.1 software and later releases, the WMT streaming module contains two sets of processes that handle client requests:

- The mms_server processes that handle MMS-over-HTTP
- The mcast_mms processes that handle MMS requests over IP multicast

In the ACNS 5.2.1 software and later releases, standalone Content Engines use the Fast Start and Fast Cache features to stream live stream-split content or on-demand (cache-hit) content to the client; a Windows Media 9 server cannot stream content to a Content Engine using Fast Start or Fast Cache.

About WMT Streaming and Caching Services with Standalone Content Engines

The term WMT streaming and caching services is used to refer collectively to the two groups of WMT services:

- WMT MMS services (supported in the ACNS 5.2.1 software to ACNS 5.4.x software releases)
- WMT RTSP services for WMS 9 clients and servers (supported in the ACNS 5.3.1 software and later releases)

Table 9-2 lists the types of WMT streaming and caching services that are supported with standalone Content Engines that are running the ACNS 5.2.1 software and later releases.
Overview of the WMT Streaming and Caching Services

Table 9-2  WMT Streaming and Caching Services with Standalone Content Engines

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMT proxy caching</td>
<td>The Content Engine receives WMT requests directly from a Windows Media player. The Content Engine retrieves the requested content if it is not already stored in its local cache, stores a copy locally whenever possible, and sends the requested content to the client. For more information, see the “About WMT Proxy Caching” section on page 9-6.</td>
</tr>
<tr>
<td>WMT transparent caching</td>
<td>The Content Engine receives WMT requests that are transparently redirected to it by a WCCP Version 2 router or a Layer 4 switch. The Content Engine retrieves the requested content if it is not already stored in its local cache, stores a copy locally whenever possible, and sends the requested content to the client. For more information, see the “About WMT Transparent Caching” section on page 9-7.</td>
</tr>
<tr>
<td>Distribution of WMT live streams (common)</td>
<td>The Content Engine serves WMT live streams to all local users (Windows Media players) whose WMT traffic it receives. The WMT live streams can be unicast or multicast live feeds. The Content Engine splits the live feeds into multICAST or unicast to relay the stream to the WMT client. For more information, see the “About Live Splitting with WMT” section on page 9-8.</td>
</tr>
<tr>
<td>Distribution of preloaded VOD files (rare)</td>
<td>VOD files are preloaded on the Content Engine for on-demand delivery of these files to the Windows Media players. VOD caching is similar to HTTP caching; however, VOD files are cached in a different file system (mediafs) on the standalone Content Engine. To configure a standalone Content Engine to distribute VOD files, follow these steps: 1. Preload the VOD files on this Content Engine, as described in the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2. 2. Publish the URLs of the preloaded VOD files that clients can now access through their WMT media players.</td>
</tr>
</tbody>
</table>

About Caching Policies in WMT Streaming Media Caching

In contrast to HTTP caching, caching policies in WMT streaming media caching are much simpler, because streaming media is mostly large static content. The caching policy in WMT caching is straightforward. All responses are cacheable, including partial responses. All WMT requests result in communication between the Content Engine and the origin server, even if the request is a cache hit.

By establishing the streaming control session, the Content Engine can verify that its cached content is fresh, and the client can access the content. Because streaming objects are typically very large in size, the overhead of establishing the control session with the server is minimal and does not reduce the bandwidth savings from the cache hits.

About WMT Proxy Caching

If direct proxy routing is being used to direct WMT requests to the standalone Content Engine, then you can configure the Content Engine to support WMT proxy caching. In direct proxy mode, the standalone WMT-enabled Content Engine accepts incoming WMT streaming requests directly from WMT clients.
Overview of the WMT Streaming and Caching Services

(end users who are using the Windows Media player to request WMT content) and acts on behalf of these clients, communicating with the origin WMT server. This type of caching is referred to as WMT proxy caching.

If the client is a Windows Media 6.4 or 7.0 player, the Content Engine accepts and serves the streaming requests over MMS-over-HTTP. (See Figure 9-2.)

**Figure 9-2  WMT MMS Proxy Caching (Direct Proxy Routing)**

If the client is a Windows Media 9 player and the Content Engine is running the ACNS 5.3.1 software and later releases, the Content Engine can accept and serve the streaming request over RTSP and HTTP. (See Figure 9-1.)

**Note**

If a firewall is positioned between a Content Engine and a requesting client, make sure that you assign the external IP address of the Content Engine when explicitly configuring the Windows Media player proxy settings on the end users’ desktops to point to directly to this Content Engine.

You can use the Setup utility or the Content Engine CLI to enable and configure WMT proxy caching on a standalone Content Engine that is running the ACNS 5.2.1 software and later releases.

- For information about how to use the Setup utility to configure WMT proxy caching on a standalone Content Engine, see the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21.
- For information about how to use the Content Engine CLI to configure WMT proxy caching, see the “Enabling and Configuring WMT Caching on Standalone Content Engines” section on page 9-34.

**About WMT Transparent Caching**

If WMT transparent redirection (WCCP or Layer 4 switching) is being used to direct WMT requests to a standalone Content Engine, then you can configure the Content Engine to support WMT transparent caching. In this case, the standalone Content Engine is acting as a transparent proxy server for clients who are requesting WMT content, and the Content Engine is not visible to these clients. After receiving a transparently redirected WMT request, the Content Engine retrieves the requested content if it is not already stored in its local cache, stores a copy locally whenever possible, and sends the requested content to the client media player.

You can use the Setup utility or the Content Engine CLI to enable and configure WMT transparent caching on a standalone Content Engine that is running the ACNS 5.2.1 software and later releases. If you use the Setup utility, you can only configure the Content Engine to accept redirected WMT requests from WCCP Version 2 routers. If you use the Content Engine CLI, you can configure the Content Engine to accept redirected WMT requests from Layer 4 switches as well as from WCCP Version 2 routers.
Overview of the WMT Streaming and Caching Services

For information about how to use the Setup utility to configure WMT transparent caching on a standalone Content Engine, see the “Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine” section on page 4-21. For information about how to configure WMT transparent caching on a standalone Content Engine through the Content Engine CLI, see the “Enabling and Configuring WMT Transparent Caching on Standalone Content Engines” section on page 9-35.

About Live Splitting with WMT

The WMT-enabled Content Engine also supports live splitting. By splitting requests for live streams. A single stream from the origin streaming server is split to serve each client that requested the stream. (See Figure 9-3.) If a WMT client requests a publishing point on a remote streaming server without specifying an ASF file, the Content Engine dynamically creates an alias file that references the remote streaming server. All further requests to that remote streaming server are served by having the Content Engine split the stream and serve it to the WMT clients.

When the first client (Client 1) that requested the original stream disconnects from the network, the Content Engine continues to serve the other clients (Client 2 and Client 3), until all clients disconnect from the network.

Figure 9-3  How a Standalone Content Engine Supports Live Splitting

By having the Content Engine perform the live splitting, you potentially save considerable network bandwidth between the client and the origin streaming server because the Content Engine is closer to the clients.

Live splitting is supported for different data packet transport protocols (HTTP and RTSP). In the ACNS 5.3.1 software and later releases, live splitting for two additional transport protocols, RTSPU and RTSPT, is available. To display aggregated statistics about WMT live streams, enter the show statistics wmt streamstat live EXEC command.
About Proxy Authentication for a WMT-Enabled Content Engine

The WMT-enabled Content Engine supports both basic and NTLM authentication by the origin server. When a client requests content that needs user authentication, the Content Engine 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.
Chapter 9 Configuring WMT Streaming Media Services on Standalone Content Engines

Configuration Guidelines

The following are the three types of proxy authentication methods:

- **Basic authentication**—An authentication scheme in which 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 nontransparent proxy mode (direct proxy routing) as well as transparent proxy mode, over HTTP.

- **Windows NTLM authentication**—A connection-based challenge-response authentication scheme. Because the NTLM protocol authenticates every connection, the proxy cannot arbitrarily create new connections with the origin server, and the proxy must reuse connections initiated by the client.

  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. NTLM authentication is supported in nontransparent proxy mode (direct proxy routing) as well as transparent proxy mode, over HTTP. The proxy supports caching and delivery of Digital Rights Management (DRM)-protected Windows Media files. Access control lists (ACLs) enforced by the origin server are automatically enforced by the proxy.

- **Microsoft Digest authentication**—An authentication method in which an initial authentication of the client is performed when the server receives the first challenge response from the client. After the server verifies that the client has not been authenticated yet, it accesses the services of a domain controller (DC) to perform the initial authentication of the client. When the initial authentication of the client is successfully completed, the server receives a Digest session key. The server caches the session key and uses it to authenticate subsequent requests for resources from the authenticated client. This is a connection-based challenge-response authentication scheme similar to NTLM authentication. This authentication scheme is supported in nontransparent proxy mode (direct proxy routing) as well as transparent proxy mode over HTTP.

  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 5.1 software and later releases.

  In the ACNS 5.3.1 software and later releases, pass-through authentication support for WMT RTSP requests from Windows Media 9 players is available. For more information on this topic, see the “Configuring Pass-Through Authentication for WMT Requests” section on page 10-7.

Configuration Guidelines

This section provides some general guidelines for configuring Windows media streaming and caching services on standalone Content Engines, and then provides instructions for configuring describes how to configure Windows Media services for standalone Content Engines.

When configuring Windows Media streaming and caching services with standalone Content Engines, note the following important points:

- Windows Media Services 9 Series 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 9 players), the server and distribution application (Windows Media 9 server) and the encoder application (Windows Media Encoder).
• If the WMT proxy server fails to serve a request that uses MMS-over-HTTP, the Windows Media 9 player will bypass the proxy and serve the request from the origin server. Previous versions of the Windows Media players (Version 6.4 and 7.0) did not support this feature. Typically, proxy servers fail to serve a request for one of these reasons:
  – The requested media file exceeds the configured values in the Content Engine (bandwidth, maximum number of sessions, or maximum bit rate).
  – The URL fails to comply with the rules or URL filter configured in the Content Engine.
  – The proxy server is down.

Table 9-3 lists the supporting WMT incoming and outgoing proxy modes. As this table indicates, the modes vary based on the release of the ACNS 5.x software that is running on the Content Engine.

<table>
<thead>
<tr>
<th>Proxy Mode</th>
<th>ACNS 5.3.1 Software or Later</th>
<th>ACNS 5.2.1 Software or Earlier</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incoming proxy mode</td>
<td></td>
<td></td>
<td>For more information about WMT RTSP transparent redirection, see the “Configuring RTSP Transparent Redirection of WMT Requests” section on page 9-31.</td>
</tr>
<tr>
<td></td>
<td>• MMS transparent proxy (WCCP transparent redirection through services 81 and 82).</td>
<td>• WMT transparent proxy (WCCP transparent redirection through service 82 and 83)</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>ACNS 5.5.x software does not support MMS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• WMT RTSP transparent proxy (WCCP transparent redirection through services 80 and 83)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outgoing proxy mode</td>
<td>• MMS-over-HTTP proxy mode</td>
<td>• MMS-over-HTTP proxy mode</td>
<td>Use the <code>wmt proxy outgoing</code> global configuration command to configure a WMT outgoing proxy on the Content Engine. In the ACNS 5.3.1 software and later releases, support for an RTSP outgoing proxy server is also available (the <code>wmt proxy outgoing rtsp host</code> command) for WMT RTSP requests from Windows Media 9 players. For examples of how to configure an outgoing proxy server, see Step 10 of the “Configuring General WMT Settings on Standalone Content Engines” section on page 9-18</td>
</tr>
<tr>
<td></td>
<td>• RTSP proxy mode</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
You can configure numerous WMT features with the `wmt` global configuration command.

ContentEngine(config)# wmt ?
   accelerate                   WMT streaming acceleration
   accept-license-agreement     Accept license; View by 'show wmt license-agreement'
   advanced                    WMT advanced configuration
   bandwidth                   WMT bandwidth configurations
   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.
   extended                    WMT extended configurations
   fast-cache                  Fast-cache feature
   fast-start                  Fast-start feature
   http                        MMS over HTTP configurations
   incoming                    Configuration for incoming WMT requests
   l4-switch                   Configure layer-4 switch interoperability
   license-key                 Required license key for WMT
   live-url-stripping          Strip live URL's ? and beyond
   max-concurrent-sessions     Maximum number of unicast clients that can be served concurrently.
   multicast                   Multicast configuration and scheduling.
   proxy                       Out-going proxy configuration
   transaction-logs            WMT transaction log configuration

For an example of how to use the `wmt` global configuration command to configure WMT general settings, see the “Configuring General WMT Settings on Standalone Content Engines” section on page 9-18.

WMT Proxy Server Requirements

The following are requirements for a standalone Content Engine that will be functioning as a WMT proxy server:

- Interoperability is the most important requirement for WMT software components. The WMT proxy server is required to work with all versions of Microsoft Windows Media player, Windows Media Encoder, and third-party Windows Media applications.
- In order to support WMT transparent caching, WCCP Version 2 must be running on the standalone Content Engine.
- You must configure disk space to include mediafs storage with the `disk config` command before you can cache streaming media using WMT.
- The mediafs partitions is mounted on the standalone Content Engine. This is the storage partition that is used to store any WMT streaming media content that is cached on the Content Engine.
- The Content Engine is running the ACNS 5.2.1 software and later releases.
- You have a Microsoft WMT license key. The Microsoft WMT product is licensed software. To enable the licensed WMT product feature on a standalone Content Engine, you must have a WMT license key, which is supplied on a certificate shipped with the Content Engine. For information about how to specify the WMT license key, see the “Enabling WMT Licenses on Standalone Content Engines” section on page 9-17.

Note: If you are downloading the ACNS 5.x software, you can purchase a WMT license through the Cisco.com website.
If the WMT license key is no longer needed on the Content Engine because the WMT licensed product feature is not needed, you can uninstall the WMT license key by entering the `no wmt license-key` global configuration command. After a license key is uninstalled on one Content Engine, it can be used on another device if that device supports the WMT license key.

**Note** You must disable the WMT feature using the `no wmt enable` global configuration command before uninstalling the WMT license key on a standalone Content Engine.

- You have the IP address of the standalone Content Engine that will be configured as a WMT proxy server.
- You have the IP address of the WCCP Version 2-enabled routers if you want to use transparent WCCP redirection.

### Checklist for Configuring WMT Streaming and Caching Services on Standalone Content Engines

Table 9-4 is a checklist of tasks for configuring WMT streaming and caching services on standalone Content Engines that are running the ACNS 5.2.1 software and later releases. This checklist includes the steps involved in configuring these services on a standalone Content Engine, as well as how to configure how WMT requests are routed to this standalone Content Engine.

**Note** The Setup utility allows you to enable WMT on a standalone Content Engine that is running the ACNS 5.2.1 software and later releases, and then configure WMT proxy caching and WMT transparent caching on the Content Engine. For information on this topic, see the “Configuring a Basic Configuration on Standalone Content Engines with the Setup Utility” section on page 4-10.

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
</table>
| 1.   | Enable WMT on the standalone Content Engine.  
|      | a. Accept the WMT license agreement.  
|      | b. Accept the evaluation WMT license, or specify your Cisco permanent WMT license.  
|      | c. Enable the licensed WMT feature on the standalone Content Engine.  
|      | See the “Enabling WMT Licenses on Standalone Content Engines” section on page 9-17.  
| 2.   | Configure one or more of the following routing methods to direct client requests for Windows Media content to this standalone Content Engine:  
|      | – Direct proxy routing (nontransparent)  
|      | – Transparent redirection (WCCP Version 2 routing or Layer 4 switching)  
|      | With direct proxy routing, the Windows Media players send their WMT requests directly to this Content Engine (nontransparent forward proxy server). With direct proxy routing, you must point the Windows Media players directly to the Content Engine, as described in the “Pointing Windows Media Players Directly to a Standalone Content Engine for WMT MMS Requests” section on page 4-45. |
Chapter 9  Configuring WMT Streaming Media Services on Standalone Content Engines

Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines

This section describes how to use the Content Engine CLI to configure WMT RTSP streaming and caching services on a standalone Content Engine that is running the ACNS 5.3.1 software and later releases.

Figure 9-4 provides a detailed view on how to configure these services initially for standalone Content Engines. Table 9-4 provides a checklist of tasks for configuring these services on a standalone Content Engine.

This section also describes how to perform the necessary configuration changes to the Windows Media 9 players (if direct proxy routing is to be used) and the necessary configuration changes to WCCP Version 2 routers (if WMT RTSP redirection through WCCP [services 80 and 83] will be used).

Table 9-4  Checklist for Configuring WMT MMS Services with Standalone Content Engines  (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. For direct proxy routing, enable and configure WMT proxy caching on the Content Engine.</td>
<td>See the “Enabling and Configuring Nontransparent WMT Proxy Caching on Standalone Content Engines” section on page 9-34.</td>
</tr>
<tr>
<td>4. For transparent redirection, enable and configure WMT transparent caching on the Content Engine.</td>
<td>See the “Enabling and Configuring WMT Transparent Caching on Standalone Content Engines” section on page 9-35.</td>
</tr>
</tbody>
</table>
| 5. Choose which types of WMT streaming content that this standalone Content Engine will be distributing to clients:  
  – Video-on-demand (VOD) files  
  – Live WMT streams | For VOD files, see the “Configuring Standalone Content Engines to Distribute VOD Files” section on page 9-35.  
For live WMT streams, choose which methods this Content Engine will use to relay live WMT streams to Windows Media clients:  
  • If multicast out will be used, then go to task 6.  
  or  
  • If unicast out will be used, then go to task 7. |
| 6. Configure the Content Engine to relay live content to Windows Media clients through multicasting. | See the “Configuring Standalone Content Engines to Deliver WMT Live Streams” section on page 9-37. |
| 7. Configure the Content Engine to relay live content to Windows Media clients through unicast. | See the “Configuring Multicast-In Unicast-Out on Standalone Content Engines” section on page 9-43.  
See the “Configuring Unicast-In Unicast-Out on Standalone Content Engines” section on page 9-44. |
Figure 9-4 Configuring WMT RTSP Streaming and Caching Services with Standalone Content Engines

Begin configuration of WMT RTSP caching and streaming services for a standalone Content Engine

Use the Setup utility or Content Engine CLI to enable WMT on the standalone Content Engine
- Accept the WMT license agreement
- Specify a WMT license key (accept the evaluation WMT license key, or enter your permanent key)
- Turn on the WMT feature on this Content Engine

Use the Content Engine CLI to specify the following settings for the RTSP gateway (the single point of entry for RTSP messages) that runs on the standalone Content Engine and is automatically enabled
- Specify the IP address of the RTSP gateway if the Content Engine is behind a NAT-enabled router (required)
- Change the default basic settings, such as the RTSP incoming port number (optional)
- Change the default advanced settings, such as the maximum request rate (optional)

Configure one or more of the following routing methods to direct RTSP requests from Windows Media 9 players to this standalone Content Engine

- **WCCP routing or Layer 4 switching**
  - RTSP transparent redirection of WMT requests
  - Configure WMT RTSP redirection through WCCP services 80 and 83
    - Configure on WCCP routers that will support this media service
    - Configure on this Content Engine
- **Direct proxy routing (nontransparent)**
  - Configure RTSP redirection through Layer 4 switching
    - Configure on Layer 4 switch
    - Configure on this Content Engine
  - Manually configure Windows Media 9 players to point directly to this Content Engine (nontransparent forward proxy server)

- **Transparent caching**
  - caching VOD files) and live splitting with this Content Engine (a Windows Media 9 server)
  - Configure WMT RTSP streaming services on this Content Engine
    - for example, configure fast start, fast cache, and live splitting
  - Configuration of WMT RTSP streaming and caching services for a standalone Content Engine is complete
After you have configured the WMT RTSP caching services as depicted in Figure 9-4, you use the exact same procedure to configure WMT streaming services regardless of whether the MMS-over-HTTP or the RTPS protocol will be used to deliver the WMT streaming content (live WMT streams and VOD files) to the WMT clients.

Checklist for Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines

Table 9-5 is a checklist of tasks for configuring WMT RTSP streaming and caching services on standalone Content Engines that are running the ACNS 5.3.1 software and later releases. This checklist includes the steps involved in configuring these services on a standalone Content Engine, as well as how to configure how the RTSP requests from Windows Media 9 players are routed to this standalone Content Engine that is functioning as a Windows Media 9 server.

Table 9-5  Checklist for Configuring WMT RTSP Services with Standalone Content Engines and Windows Media 9 Players

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enable WMT on the standalone Content Engine.</td>
<td></td>
</tr>
<tr>
<td>a. Accept the WMT license agreement.</td>
<td></td>
</tr>
<tr>
<td>b. Accept the evaluation WMT license, or specify your Cisco permanent WMT license.</td>
<td></td>
</tr>
<tr>
<td>c. Enable the licensed WMT feature on the standalone Content Engine.</td>
<td>See the “Enabling WMT Licenses on Standalone Content Engines” section on page 9-17.</td>
</tr>
<tr>
<td>2. If necessary, specify the RTSP gateway settings.</td>
<td></td>
</tr>
<tr>
<td>a. If the Content Engine is behind a NAT-enabled router, you must specify the IP address of the RTSP gateway (required).</td>
<td></td>
</tr>
<tr>
<td>b. You can also change the default basic and advanced RTSP gateway settings (optional).</td>
<td>See the “Configuring the RTSP Gateway for Standalone Content Engines” section on page 8-14.</td>
</tr>
<tr>
<td>3. Configure one or more of the following routing methods to direct RTSP requests from Windows Media 9 players to this standalone Content Engine:</td>
<td></td>
</tr>
<tr>
<td>– Direct proxy routing (nontransparent)</td>
<td></td>
</tr>
<tr>
<td>– WMT RTSP transparent redirection (WCCP Version 2 routing or Layer 4 switching)</td>
<td>With direct proxy routing, the Windows Media 9 players send their WMT RTSP requests directly to this Content Engine (nontransparent forward proxy server). With direct proxy routing, you must point the Windows Media 9 players directly to the Content Engine, as described in the “Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests” section on page 4-43. With WCCP routing or Layer 4 switching, you must configure the WCCP routers or Layer 4 switches and the Content Engine (transparent proxy server) for WMT RTSP transparent redirection, as described in the “Configuring RTSP Transparent Redirection of WMT Requests” section on page 9-31.</td>
</tr>
<tr>
<td>4. For direct proxy routing, enable and configure WMT proxy caching on the Content Engine.</td>
<td>See the “Enabling and Configuring Nontransparent WMT Proxy Caching on Standalone Content Engines” section on page 9-34.</td>
</tr>
<tr>
<td>5. For transparent redirection, enable and configure WMT transparent caching on the Content Engine.</td>
<td>See the “Enabling and Configuring WMT Transparent Caching on Standalone Content Engines” section on page 9-35.</td>
</tr>
</tbody>
</table>
Enabling WMT Licenses on Standalone Content Engines

Before enabling licenses for WMS on a Content Engine, make sure that your Content Engine clock and calendar settings are correct; otherwise, you will see an error message and the services will fail to install. To display the system clock, use the `show clock` EXEC command. To set the system clock, use the `clock set` EXEC command.

To use the Content Engine CLI to enable Windows Media Services on a standalone Content Engine, follow these steps:

**Step 1** View the WMT license agreement.

```
ContentEngine# show wmt license-agreement
```

**Step 2** After reading the license agreement, enter global configuration mode and accept the license agreement.

```
ContentEngine# configure terminal
ContentEngine(config)# wmt accept-license-agreement
```

**Step 3** Enter your Cisco license key for the WMT product.

```
ContentEngine(config)# wmt license-key licensekey
```

Alternatively, accept an evaluation WMT license.

```
ContentEngine(config)# wmt evaluate
```

**Step 4** Enable the WMT feature on this Content Engine.

```
ContentEngine(config)# wmt enable
```

Table 9-5 Checklist for Configuring WMT RTSP Services with Standalone Content Engines and Windows Media 9 Players (continued)

<table>
<thead>
<tr>
<th>Task</th>
<th>Additional Information and Instructions</th>
</tr>
</thead>
</table>
| 6. Choose which types of WMT streaming content that this standalone Content Engine will be distributing to clients:  
  - Video-on-demand (VOD) files  
  - Live WMT streams | For VOD files, see the “Configuring Standalone Content Engines to Distribute VOD Files” section on page 9-35.  
For live WMT streams, choose which methods this Content Engine will use to relay live WMT streams to Windows Media clients:  
  - If multicast out will be used, then go to task 7.  
  - If unicast out will be used, then go to task 8. |
| 7. Configure the Content Engine to relay live content to Windows Media 9 players through multicasting. | See the “Configuring Standalone Content Engines to Deliver WMT Live Streams” section on page 9-37. |
| 8. Configure the Content Engine to relay live content to Windows Media 9 players through unicast. | See the “Configuring Multicast-In Unicast-Out on Standalone Content Engines” section on page 9-43.  
See the “Configuring Unicast-In Unicast-Out on Standalone Content Engines” section on page 9-44. |
Step 5 When asked if you want to proceed, enter yes to proceed.

When asked if you want to proceed, enter yes to proceed.

This operation needs to restart http proxy and real proxy (if running) for memory reconfiguration. Proceed? [no] yes

The next step is to choose one or more of the following routing methods to direct client requests for Windows Media content to this standalone Content Engine:

- WCCP routing or Layer 4 switch (WMT transparent redirection for WMT MMS requests, and WMT RTSP transparent redirection for WMT RTSP requests)
- Direct proxy routing (nontransparent)

With direct proxy routing, the Windows Media players send their requests directly to this Content Engine (acting as a nontransparent forward proxy server). For instructions on how to configure a Windows Media player on the end user desktops to point directly to this Content Engine as their proxy server, see the “Pointing Windows Media Players Directly to a Standalone Content Engine for WMT MMS Requests” section on page 4-45.

Configuring General WMT Settings on Standalone Content Engines

To configure the general WMT settings on a standalone Content Engine through the Content Engine CLI, follow these steps:

Step 1 Disallow specific WMT client protocols for streaming with the wmt disallowed-client-protocols global configuration command:

\[
\text{wmt disallowed-client-protocols \{http | rtspt | rtspu\}}
\]

In the ACNS 5.3.1 software release, the following changes were made to this command:

- The rtspt and rtspu options were added.
- The tcp and udp options are hidden for backward compatibility.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disallowed-client-protocols</td>
<td>Specifies disallowed WMT client protocols.</td>
</tr>
<tr>
<td>http</td>
<td>Disallows streaming over the HTTP protocol (http://).</td>
</tr>
<tr>
<td>rtspt</td>
<td>Disallows streaming over the rtspt protocol (rtspt://).</td>
</tr>
<tr>
<td>rtspu</td>
<td>Disallows streaming over the rtspu protocol (rtspu://).</td>
</tr>
</tbody>
</table>

Step 2 Configure the maximum number of unicast clients that a standalone Content Engine can support concurrently. The default is 2500 clients.

\[
\text{ContentEngine(config)# wmt max-concurrent-sessions number}
\]

number specifies the maximum number of incoming unicast requests that the Content Engine should serve concurrently. This limit is subject to physical resources on the Content Engine (1 to 8000).

Step 3 Specify the maximum bandwidth for preloading WMT content on the Content Engine.
With the ACNS 5.x software, you can preload WMT streaming media files that may have different bit rates at the URL specified for content preloading. You can also control WMT bandwidth and bit rates using the `wmt max-bandwidth` and `wmt max-bitrate` global configuration commands.

**Step 4** Specify the maximum bit rate per WMT stream that can be received by the Content Engine.

By default, there is no limit. This bit rate is called the **WMT incoming stream bitrate**.

```
ContentEngine(config)# wmt bitrate wmt incoming bitrate
```

`bitrate` specifies the WMT incoming stream bit rate in kilobits per second. This value can be from 0 to 2,147,483,647.

**Step 5** Specify the maximum bit rate per WMT stream that can be served by the Content Engine. By default, there is no limit. This bit rate is called the **WMT outgoing stream bit rate**.

```
ContentEngine(config)# wmt bitrate wmt incoming bitrate
```

`bitrate` specifies the WMT outgoing stream bit rate in kilobits per second. This value can be from 0 to 2,147,483,647.

For more information about configuring bandwidth and bit rates, see the “Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates” section on page 9-23.

**Step 6** Specify the maximum size of a single object that the Content Engine should store in its WMT cache.

Use the `wmt cache max-obj-size` global configuration command to specify this value. The range of values is between 1 and 1,000,000 megabytes. The default value is 1024 megabytes.

```
ContentEngine(config)# wmt cache max-obj-size size
```

**Step 7** Enable WMT caching on the standalone Content Engine if it is not already enabled.

```
ContentEngine(config)# wmt cache enable
```

**Step 8** Enable WMT live URL stripping.

```
ContentEngine(config)# wmt live-url-stripping enable
```

**Step 9** Enable transparent redirection through a Layer 4 switch instead of through a WCCP router:

- a. On the Content Engine, enable transparent redirection of MMS requests through a Layer 4 switch.

  ```
  ContentEngine(config)# wmt l4-switch enable
  ```

- b. On the Content Engine, enable transparent redirection of RTSP requests through a Layer 4 switch.

  ```
  ContentEngine(config)# rtsp l4-switch enable
  ```

**Step 10** To configure a WMT outgoing proxy on the Content Engine, use the `wmt proxy outgoing` global configuration command.

Configure a Content Engine to send all of its MMS cache miss traffic to a specific MMS outgoing proxy server, or send its MMS-over-HTTP miss traffic to a specific HTTP outgoing proxy server, or send all of its WMT RTSP cache miss requests to a specific RTSP outgoing proxy server without using ICP or WCCP. In the ACNS 5.3.1 software and later releases, you can also configure an RTSP outgoing proxy server for WMT RTSP requests from Windows Media 9 players.

The command syntax is as follows:

```
wmt proxy outgoing {http | mms | rtsp} host {hostname | ip-address}
```
where:

- **http** is the keyword for an outgoing MMS-over-HTTP proxy configuration.
- **rtsp** is the keyword for an outgoing RTSP proxy configuration.
- **host** is the keyword for the outgoing proxy server.
- **hostname** is the hostname or ip-address is the IP address of the outgoing proxy server.

In the following example, a Content Engine at a branch office is configured to use MMS-over-HTTP to send all its WMT cache miss requests to a central Content Engine at 172.16.30.30 through port 8080:

```plaintext
ContentEngine(config)# wmt proxy outgoing http host 172.16.30.30 8080
```

In the following example, a Content Engine at a branch office is configured to send all its MMS cache miss requests to a central Content Engine at 172.16.30.31 through port 1700:

```plaintext
ContentEngine(config)# wmt proxy outgoing http host 172.16.30.31 1700
```

In the following example, the Content Engine at a branch office is configured to use MMS-over-TCP (MMST) or MMS-over-UDP (MMSU) to send all its cache miss RTSP requests to a central Content Engine at 172.16.30.30 through port 8080:

```plaintext
ContentEngine(config)# wmt proxy outgoing rtsp host 172.16.30.30 8080
```

In the following example, the Content Engine at a branch office is configured to use RTSP to send all its cache miss RTSP requests from Windows Media 9 players to a central Content Engine at 172.16.30.30 through port 8080:

```plaintext
ContentEngine(config)# wmt proxy outgoing rtsp host 172.16.30.30 8080
```

**Step 11** Decide which type of media file should be served by WMT.

Typically, Content Engines are shipped with a default list of filename extensions to be served by WMT. The default list in the Content Engine contains the following filename extensions: asf, none, nsc, wma, and wmv. The default list of filename extensions includes none in order to enable a Content Engine to serve media files without file extensions (for example, broadcast aliases or URLs of live encoders). The filename extension nsc is included in the list to enable a Content Engine to multicast media files.

a. To add filename extensions to this list, use the **wmt http allow extension** global configuration command.

b. To remove a filename extension from the list, use the **no wmt http allow extension** global configuration command.

**Note** In the ACNS 5.2.1 software release, the **wmt mms allow extension** EXEC command was replaced with the **wmt http allow extension** EXEC command. The **show wmt mms allow extension** EXEC command was also replaced with the **show wmt http allow extension** EXEC command.

The following restrictions apply to adding new file extensions to the list:

- You cannot have more than 20 extensions in the list of allowed file extensions.
- File extensions must be alphanumeric, and the first character of every extension should be an alphabetic one.
- You cannot have more than 10 characters in a file extension.
Chapter 9  Configuring WMT Streaming Media Services on Standalone Content Engines

Configuring General WMT Settings on Standalone Content Engines

The following example adds the file extension mp3 to the list of file extensions to be served by WMT:

```
ContentEngine(config)# wmt http allow extension mp3
ContentEngine(config)#
```

**Step 12**  View the file extensions included in the list after you add or delete file extensions.

```
ContentEngine(config)# exit
ContentEngine# show wmt http allow extension
```

The `show wmt http allow extension` EXEC command does not display anything if you have not modified the default list.

**Step 13** (Optional) Disable one or more of the following three WMT streaming acceleration features that by default are enabled on the Content Engine:

- Live split
- Proxy-cache
- VOD

Use the appropriate `no wmt accelerate` global configuration command to disable the feature.

a. To disable the acceleration of live splitting, enter the `no wmt accelerate live-split` command. To reenable this feature on the Content Engine, enter the `wmt accelerate live-split enable` command.

b. To disable the acceleration of proxy caching, enter the `no wmt accelerate proxy-cache` command. To reenable this feature on the Content Engine, enter the `wmt accelerate proxy-cache enable` command.

c. To disable the acceleration of serving VOD files to WMT clients, enter the `no wmt accelerate vod` command. To reenable this feature on the Content Engine, enter the `wmt accelerate VOD enable` command.

**Step 14**  Configure the WMT Fast Streaming features on the Content Engine:

a. Specify the maximum burst bandwidth (in kilobits per second [kbps]) for the Fast Start feature. This value specifies the maximum burst bandwidth that a single player can use for accelerated initial buffering of the streaming content. For example:

```
ContentEngine(config)# wmt fast-start max-bandwidth 3000
```

The Fast Start feature allows the Windows Media 9 server to push the beginning portions of a stream to the Windows Media 9 player at the maximum available bandwidth. This feature is enabled on a Content Engine by default. The increased bandwidth that this feature initially uses to send data to the Windows Media 9 player can overburden a network if many players connect to the stream at the same time. The maximum burst bandwidth can be from 1 to 65535 kbps. The default is 3600. The maximum value is associated with the WMT license. By default, the Fast Start feature is enabled on the Content Engine. For more information, see the “Configuring Fast Start on Standalone Content Engines” section on page 9-28.

b. Specify the maximum delivery rate (maximum acceleration factor) for the Fast Cache feature. For example:

```
ContentEngine(config)# wmt fast-cache max-delivery-rate 5
```

The Fast Cache feature allows the stream rendering rate to be decoupled from the stream delivery rate on the network. This allows a Windows Media 9 server to send the stream content faster than the client’s rendering speed. The maximum delivery rate (that is, the Fast-Cache speed multiplier) can be from 1 to 65535. (Setting the `max-delivery-rate` value to 1 is equivalent to disabling the Fast
Chapter 9  Configuring WMT Streaming Media Services on Standalone Content Engines

Configuring General WMT Settings on Standalone Content Engines

Cache feature.) By default, the Fast Cache feature is enabled on the Content Engine. For more information, see the “Configuring Fast Cache on Standalone Content Engines” section on page 9-29.

**Step 15** Configure the WMT advanced client features on the Content Engine:

a. Specify the client maximum packet size (WMT maximum IP packet size) in bytes. The value can be from 512 to 2048 bytes. The default is 1500 bytes. For example:

```
ContentEngine(config)# wmt advanced client maximum-packet-size 1800
```

b. Specify the maximum amount of time that the Content Engine is to wait for a response from a WMT client before timing out the connection. The value can be from 30 to 300 seconds. The default timeout is 120 seconds. For example:

```
ContentEngine(config)# wmt advanced client idle-timeout 100
```

**Note** These two `wmt advanced client` global configuration commands are available in the ACNS 5.3.1 software and later releases.

**Step 16** Decide whether you want this Content Engine to forward its WMT logs to the upstream server (that is, a Windows Media server or another Content Engine.)

By default, Content Engines that are running the ACNS 5.3.1 software and later releases will forward their WMT logs to the upstream server. This feature applies to all of the supported protocols (for example, HTTP and RTSP [RTSPT and RTPU]).

To disable this feature and configure the Content Engine to not forward its WMT logs to the upstream server, enter the `no wmt advanced server log-forwarding enable` global configuration command. To reenable this feature, enter the `wmt advanced server log-forwarding enable` global configuration command.

**Step 17** Configure WMT logging on the Content Engine. For more information, see the “Using WMT Logging with Standalone Content Engines” section on page 9-47.

**Step 18** (Optional) Preload VOD files on the Content Engine for on-demand delivery of these files to Windows Media clients. For more information, see the “Configuring Standalone Content Engines to Distribute VOD Files” section on page 9-35.

**Step 19** (Optional) Configure the standalone Content Engine to deliver WMT live streams to the WMT clients. For more information, see the “Configuring Standalone Content Engines to Deliver WMT Live Streams” section on page 9-37.

**Step 20** Display statistics about WMT requests that are being serviced by this Content Engine. For more information, see the “Displaying Information about the WMT RTSP Server Configuration” section on page 9-46.

**Step 21** Configure URL filtering for WMT requests that are directed to this Content Engine. For more information, see Chapter 11, “Configuring Content Preloading and URL Filtering on Standalone Content Engines.”

**Step 22** Configure rules for WMT requests that are directed to this Content Engine. For more information, see Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”

**Step 23** Display statistics for the configured rules for WMT requests.

For example, enter the `show statistics rtsp EXEC command to display statistics that are related to the configured RTSP and WMT RTSP rules. For more information, see the “Displaying Statistics for Configured Rules” section on page 13-33.
### Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates

The bandwidth between the WMT client and the WMT proxy server (Content Engine A) is called **outgoing WMT bandwidth**. The bandwidth between the WMT proxy server (Content Engine A) and the Windows Media 9 server is called **incoming WMT bandwidth**. (See Figure 9-5.)

**Figure 9-5   Incoming and Outgoing Bandwidth and Bit Rates**

In the ACNS 5.3.1 software and later releases, you can configure bandwidth for MMS-over-HTTP and RTSP requests.

In addition to incoming and outgoing WMT bandwidth, there are incoming and outgoing WMT bit rates per session. When a WMT client requests media files for the first time, the WMT proxy server (that is, the Content Engine) caches the on-demand Windows Media files. All subsequent requests for the same file are served by the WMT proxy server from its cache. The WMT proxy server can also live split a broadcast; it can pull only one unicast stream for the origin streaming server (for example, an external Windows Media 9 server) and live split the broadcast to multiple WMT clients.

The bandwidth between the WMT proxy and the origin streaming server is called the **incoming bandwidth**. Because the bandwidth from the edge to the outside IP WAN is very limited, it is important that you specify a per-session limit (the maximum bit rate per request) for each service that is running on the Content Engine and that consumes incoming bandwidth (for example, the WMT streaming service), as well as an aggregate limit (the maximum incoming bandwidth.) The outgoing bandwidth needs to be controlled based on the WMT license that is configured on the Content Engine.

To specify a WMT incoming and outgoing bandwidth, use the `bandwidth wmt outgoing` and `bandwidth incoming` global configuration commands:

- To specify the outgoing WMT bandwidth in kbps, use the `bandwidth wmt outgoing kbps` global configuration command. This command sets the maximum bandwidth for WMT content that can be delivered to a client that is requesting WMT content. The range of values is between 0 and 2,147,483,647 Kbps.

  If the specified outgoing bandwidth exceeds the limit specified by the WMT license then a warning message is displayed to inform you of this situation. However, the specified outgoing bandwidth setting is applied because you may have configured this setting before you enabled the initial WMT license or another WMT license that has a higher limit.
To specify the incoming WMT bandwidth in Kbps, use the `bandwidth wmt incoming kbps` global configuration command. This command sets the maximum bandwidth for WMT content that can be delivered to a Content Engine from the origin streaming server or another Content Engine in the case of a cache miss. The specified bit rate is the maximum incoming WMT per session bit rate. The range of values is between 0 and 2,147,483,647 kbps. Incoming bandwidth applies to broadcast stations, multicast station, and VOD content from the origin server in the case of a cache miss.

Incoming bandwidth applies to the following:
- VOD content from the origin server in the case of a cache miss
- Broadcast stations in which the source for the broadcast station and multicast stations is a unicast (MMS-over-HTTP or RTSP-over-RTP) or a multicast. If the source is a multicast, the specified incoming bandwidth is not applied.
- Multicast stations in which the source of the multicast station is a unicast (RTSP) or a multicast.

To specify an incoming and outgoing WMT per session bit rate, use `bitrate wmt incoming` and `bitrate wmt outgoing` global configuration commands:

- To specify the maximum incoming streaming bit rate per session that can be delivered to the WMT proxy server (a Content Engine) from the origin streaming server or another Content Engine in the case of a cache miss, use the `bitrate wmt incoming bit-rate` global configuration command. The specified bit rate is the maximum incoming WMT per session bit rate. The range of values is between 0 and 2,147,483,647 kbps. The default value is 0 (no bit rate limit).
- To set the maximum outgoing streaming bit rate per session that can be delivered to a client that is requesting WMT content, use the `bitrate wmt outgoing bit-rate` global configuration command. The specified bit rate is the maximum outgoing WMT per session bit rate. The range of values is between 0 and 2,147,483,647 kbps. The default value is 0 (no bit rate limit).

Outgoing bandwidth applies to the following:
- VOD content from the WMT proxy server on the Content Engine in the case of a cache miss.
- Broadcast stations and multicast stations that are configured on the Content Engine. The source for the broadcast station can be unicast (MMS [MMST and MMSU], MMS-over-HTTP, or RTSP-over-RTP) or multicast.

### About Variable WMT Bit Rates

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 choose a particular bit rate. 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.

---

**Note**

In the case of multiple bit rate files, the Content Engine that is acting as the WMT proxy server only retrieves the bit rate that the client has requested.

### Configuring Subnet-Based Outgoing Bandwidth

In the ACNS 5.3.1 software and later releases, you can configure IP subnet-based bandwidth control for WMT requests. This feature allows you to specify the maximum bandwidth consumption for specific client IP subnets (that is, the aggregate bandwidth for the subnet). This bandwidth control feature is supported for WMT streaming through the following protocols: Windows Media 9 RTSP and HTTP.
You specify the rules for limiting subnet-based outgoing bandwidth in an XML configuration file. This configuration file is called the advanced bandwidth configuration file. For example, you may have three subnets (Subnet A that is the parent subnet, and Subnet B and C that are within Subnet A), and you specified three subnet-based bandwidth rules as follows:

- Rule A–Subnet A, 10.1.1.0/24, has been configured with an allow bandwidth of 10000 Kbps.
- Rule B–Subnet B, 10.1.1.0/25, has been configured with an allow bandwidth of 7000 Kbps.
- Rule C–Subnet C, 10.1.1.128/25, has been configured with an allow bandwidth of 5000 Kbps.

Even though the total allowed bandwidth of Subnet B and C is 12000 kbps (as defined by Rule B and C in the configuration file), the total bandwidth will not exceed 10000 kbps because of Rule A.

The following is an example of the format of the advanced bandwidth configuration file. This example also shows the required order of the lines in the advanced bandwidth configuration file:

```
<?xml version="1.0"?>
<BandwidthSpec>
  <BandwidthRule>
    <ClientNetwork>10.77.140.133/32</ClientNetwork>
    <description>(Apply to PC jdoe-w2k)</description>
    <Allow limit="3000" service="wmt"/>
  </BandwidthRule>
  <BandwidthRule>
    <description>Comment (Apply to PCs in subnet 10.77.140.x)</description>
    <Allow limit="50000" service="wmt"/>
    <ClientNetwork>10.77.140.0/24</ClientNetwork>
  </BandwidthRule>
  <BandwidthRule>
    <Allow limit="1400" service="wmt"/>
    <ClientNetwork>10.1.1.1/32</ClientNetwork>
  </BandwidthRule>
  <Default limit="3000" service="wmt"/>
</BandwidthSpec>
```

The following information applies to the format of this advanced bandwidth configuration file:

- The <description> tag is optional.
- The <ClientNetwork> - IPAddress/Netmask entry is a required field.
- If the <Allow limit> field is specified as -1, the bandwidth allowed is unlimited.
- The Service tag currently has only one supported option (the wmt option).
- The <Default> tag is optional. This tag is used to configure the default bandwidth. If none of the subnet bandwidth rules match, the default rule is applied if it is configured.

You use FTP to download this advanced bandwidth configuration file to the Content Engine so that the file is available in the local sysfs partition on the Content Engine.

Note

The `bandwidth wmt outgoing` global configuration command configures the total outgoing WMT bandwidth, which controls the total outgoing bandwidth used for WMT streaming; regardless of any subnet-based bandwidth configuration that is specified through the advanced bandwidth configuration file.

Specify the path of the advanced bandwidth configuration file, as follows:

```
ContentEngine(config)# bandwidth advanced config-file filename-path
```
Display the WMT server bandwidth allocation statistics, as follows:

```
ContentEngine # show statistics bandwidth advanced
```

The command output shows such information as the currently used bandwidth for each specified rule in the advanced configuration file and the currently available bandwidth for each specified rule in the configuration file.

Display statistics about bandwidth allocation errors, as follows:

```
ContentEngine # show statistics bandwidth advanced errors
```

The command output shows the time the allocation error occurred, the client from which the request was received, and the amount of allocated bandwidth that was requested.

### Configuring a WMT Bandwidth Incoming Bypass List

To configure a WMT bandwidth incoming bypass list on standalone Content Engines, use the `wmt bandwidth incoming bypass-list` global configuration command.

The syntax of this command is as follows:

```
wmt bandwidth incoming bypass-list [ip-address | hostname] [ip-address | hostname]
```

Table 9-6 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandwidth</td>
<td>Configures WMT bandwidth settings on the Content Engine.</td>
</tr>
<tr>
<td>incoming</td>
<td>Allows bypassing of incoming bandwidth restrictions for broadcast alias and multicast stations.</td>
</tr>
<tr>
<td>bypass-list</td>
<td>Configures a list of up to four Content Engines that will be exempted from checking for incoming bandwidth.</td>
</tr>
<tr>
<td>ip-address</td>
<td>IP address of an exempt Content Engine.</td>
</tr>
<tr>
<td>hostname</td>
<td>Hostname of an exempt Content Engine.</td>
</tr>
</tbody>
</table>

The increased bandwidth that the Fast Start feature initially uses to send data to the Windows Media 9 player can overburden a network if many players connect to the stream at the same time. To specify the maximum burst bandwidth allowed for a single player, use the `wmt fast-start max-bandwidth number` global configuration command. For more information, see the “Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates” section on page 9-23.

### Configuring Fast Streaming Features on Standalone Content Engines

Windows Media Services 9 Series offers a set of Fast Streaming features that combine streaming, downloading, and caching features for web client acceleration improving the user experience by accelerating delivery of streaming content to the client. In versions earlier than Windows Media Services 9 Series, content was streamed at a constant bit rate to clients.
The ACNS 5.2.1 software and later releases the following Fast Streaming features are supported: Fast Start, Fast Cache, and Fast Reconnect for WMT requests. In the ACNS 5.3.1 software and later releases, this support is available for WMT RTSP requests. Table 9-7 lists the Fast Streaming features that are supported by Content Engines that are running the ACNS 5.2.1 software and later releases.

**Table 9-7**  
Fast Streaming Features Supported by Standalone Content Engines

<table>
<thead>
<tr>
<th>Feature</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Start</td>
<td>See the “Configuring Fast Start on Standalone Content Engines” section on page 9-28.</td>
</tr>
<tr>
<td>Fast Cache</td>
<td>See the “Configuring Fast Cache on Standalone Content Engines” section on page 9-29.</td>
</tr>
</tbody>
</table>

Table 9-8 lists the types of content that are supported with the Fast Start and Fast Cache features.

**Table 9-8**  
Fast Start and Fast Cache Support for Standalone Content Engines

<table>
<thead>
<tr>
<th>Feature</th>
<th>Preloaded VOD Files</th>
<th>Live Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast Start</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fast Cache</td>
<td>Yes</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

With preloaded VOD files, if the Windows Media 9 server (the Content Engine) determines that the client (a Windows Media 9 player) supports the Fast Start and Fast Cache features, it uses the Fast Start and Fast Cache features to send the packets to the clients. Otherwise, the Windows Media 9 server transmits the packets at its regular speed.

With a cache hit, the Windows Media 9 server (the Content Engine) uses the Fast Start and Fast Cache features to serve the content when the content is cached or partially cached in its local storage.

With a cache miss, the Windows Media 9 server (the Content Engine) operates as a Windows Media 7.0 player and communicates with the remote origin server. Although the origin server supports Fast Start and Fast Cache, the packets must come from the remote server and will still be at the normal speed. After the content is cached on the Content Engine (cache hit or partial hit), the Fast Start and Fast Cache features are supported over MMS (MMS-over-HTTP and RTSP-over-RTP) from Windows Media 9 players.

With live content, the Windows Media 9 server (the Content Engine) needs to hold the content in its buffer for a few seconds. When the first client requests the live stream, the buffer fills up. This buffer is used to serve Fast Start packets to subsequent clients that request the same stream. The first client does not experience any benefit from the Fast Start feature. However, the first client triggers the process of having data come into the Content Engine’s buffer. Subsequent clients who request the same live content will see the benefit of Fast Start when the Content Engine pushes the buffered content to these clients at a faster pace.

**Note**  
The Fast Start feature is only used by Windows Media 9 players that connect to a unicast stream.
A media player must fill its internal buffer (the default is 5 seconds) before it starts rendering the video. For example:

- Without the Fast Start feature, if the player’s buffer is set to 5 seconds and the stream-encoding bit rate is 100 kbps, the server will send data at 100 kbps. In this case, it takes 5 seconds for the player to fill up the internal buffer. Consequently, it takes 5 seconds before users can view the video in their players.

- With the Fast Start feature, the server can push the data at a faster pace (for example, 500 kbps). In this case, it takes only 1 second for the player to fill up its internal buffer and to start rendering the video. Consequently, it takes only 1 second before users can start to view the video in their player.

### Configuring Fast Start on Standalone Content Engines

Windows Media Services 9 Series introduced the Fast Start feature. This feature allows the server to push the beginning portions of a stream to the client at the maximum available bandwidth. This reduces the amount of time that is required to fill the player’s internal buffer, and reduces the amount of time that users (the WMT clients) need to wait before they can start to view the stream in their player (Windows Media 9 players). The Windows Media 4.1 server and Windows Media 4.1 player do not support the Fast Start feature.

**Note**
The benefit of Fast Start is not available to the first client connecting to a live stream.

In the ACNS 5.2.x software, Fast Start is only available for MMS-over-HTTP requests with Windows Media Services Version 9.0. In the ACNS 5.3 software and later releases, support for Fast Start is also available for RTSP requests from a Windows Media 9 player.

Fast Start provides the following benefits to the users:

- Better playback by eliminating buffering time, while playing a single piece of content or switching seamlessly between on-demand clips or broadcast channels

- Fast forwarding or rewinding of content without additional delay or rebuffering

- Pre-buffering of data, making the Windows Media player resistant to playback errors due to lost packets or other network problems

When Fast Start is enabled on the Content Engine, the increased bandwidth that Fast Start initially uses to send data to the media players can overburden a network if many media players connect to the stream at the same time. To reduce the risk of network congestion, use the `wmt fast-start max-bandwidth` global configuration command to limit the amount of bandwidth that Fast Start can use to stream content to each media player.

**Note**
In the ACNS 5.2.x software, the Fast Start feature is only available for MMS-over-HTTP requests with Windows Media Services Version 9.0. In the ACNS 5.3.1 software and later releases, support for the Fast Start feature is also available for RTSP requests from a Windows Media 9 player.

Standalone Content Engines, which are running the ACNS 5.2.1 software and later releases, use the Fast Start feature for preloaded video-on-demand files, cache hits, and live content. This feature is not supported for cache misses and is used only by clients that connect to a unicast stream.

Use the `wmt fast-start` global configuration command to configure the Fast Start feature on a standalone Content Engine, which is running the ACNS 5.2.1 software and later releases.
The syntax for this command is as follows:

```
wmt fast-start { enable | max-bandwidth number }
```

Table 9-9 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast-start</td>
<td>Configures the Fast Start feature.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables the Fast Start feature.</td>
</tr>
<tr>
<td>max-bandwidth</td>
<td>Maximum amount of bandwidth in kilobits per second (kbps) that a single player can use for accelerated initial buffering of the streaming content.</td>
</tr>
<tr>
<td>number</td>
<td>Maximum burst bandwidth allowed per player. The default is 3500.</td>
</tr>
</tbody>
</table>

The increased bandwidth that the Fast Start feature initially uses to send data to the player can overburden a network if many players connect to the stream at the same time. To specify the maximum burst bandwidth allowed for a single player, use the `wmt fast-start max-bandwidth number` global configuration command.

When Fast Start is enabled on the Content Engine, the increased bandwidth that Fast Start initially uses to send data to the media players can overburden a network if many media players connect to the stream at the same time. To reduce the risk of network congestion, use the `wmt fast-start max-bandwidth` global configuration command to limit the amount of bandwidth that Fast Start can use to stream content to each media player.

To configure Fast Start on a standalone Content Engine, follow these steps:

**Step 1** Enable Fast Start on the Content Engine by entering the following command:

```
ContentEngine(config)# wmt fast-start enable
```

**Step 2** Set the maximum burst bandwidth allowed per media player when Fast Start is used to serve packets to the media player by entering the following command:

```
ContentEngine(config)# wmt fast-start max-bandwidth number
```

**Step 3** Verify that Fast Start is enabled by entering the `show wmt` EXEC command.

---

**Configuring Fast Cache on Standalone Content Engines**

Windows Media Services 9 Series introduced a new feature called Fast Cache. Fast Cache allows streaming of content to the Windows Media player’s cache as fast as the network allows, reducing the possibility of an interruption in play due to network problems. When used with the Windows Media Player 9 Series, Fast Cache provides a way to stream content to clients faster than the data rate specified by the stream format. For example, with Fast Cache enabled, the server can transmit a 128-kbps stream at 700 kbps. In Windows Media player, the stream is still rendered at the specified data rate, but the media player can buffer a much larger portion of the content before rendering it. This buffering allows the client to handle variable network conditions without a perceptible impact on the playback quality of either on-demand or broadcast content.
Fast Cache is useful in the following situations:

- When the network bandwidth available to the client exceeds the bandwidth required for the content; for example, clients that use a cable modem, Digital Subscriber Line (DSL) connection, or corporate intranets.
- When the network connectivity is intermittent or has high latency; for example, wireless networks.
- When the quality of the content received is of high importance; for example, businesses that provide pay-per-view movies.

With this feature, the media data can be delivered at a rate higher than the playback rate to the client for faster delivery. With the Fast Cache feature, the stream rendering rate is decoupled from the stream delivery rate on the network. This allows a Windows Media 9 server to send the stream content faster than the client’s rendering speed. The extra data is then buffered at the Windows Media Player 9 Series client to allow the Windows Media 9 player that is running on this client to adapt better to fluctuations in network bandwidth later on.

**Note**

In the ACNS 5.2.x software, the Fast Cache feature is only available for MMS-over-HTTP requests with Windows Media Services Version 9.0. In the ACNS 5.3.1 software and later releases, support for the Fast Cache feature is also available for WMT RTSP requests from Windows Media 9 players.

The Fast Cache feature is applicable only for the RTSP and HTTP protocols.

A Windows Media 9 server informs a Windows Media 9 player that it supports the Fast Cache feature. The player then indicates to the server how fast it is for the Fast Start and Fast Cache features. When Fast Cache is configured on a Content Engine, which is running the ACNS 5.2.1 software and later releases, the Content Engine serves the content to a Windows Media 9 player using the smaller of the following two values:

- The bit rate specified in the client request
- The maximum delivery rate configured for Fast Cache in the Content Engine

The following example illustrates bandwidth control and Fast Cache speed adjustment. Client A is a client PC that is running the Windows Media 9 player, which has an IP address of 10.77.140.133, and is configured as follows:

```xml
<?xml version="1.0"?>
<BandwidthSpec>
  <BandwidthRule>
    <ClientNetwork>10.77.140.133/32</ClientNetwork>
    <description>(Apply to my PC)</description>
    <Allow limit="1100" service="wmt"/>
  </BandwidthRule>
  ...
</BandwidthSpec>
```

If Client A requests media content through RTSPT (RTSP TCP mode) and the bit rate of the requested file is 500 kbps, because the available bandwidth for Client A is 1100 kbps the Fast Cache speed for Client A is restricted to 2 (because a speed of 2 would consume 1000 Kbps [500 x 2 = 1000 kbps]). Consequently, the bandwidth consumed by Client A is less than the total available bandwidth limit of 1100 kbps.

Standalone Content Engines, which are running the ACNS 5.2.1 software and later releases, use the Fast Cache feature for preloaded video-on-demand files and for cache hits. The Fast Cache feature is not supported for cache misses, and is not applicable for the delivery of live content.
To configure Fast Cache on a standalone Content Engine, follow these steps:

---

**Step 1** Enable Fast Cache on the Content Engine by entering the following command:

```
ContentEngine(config)# wmt fast-cache enable
```

By default, the Fast Cache feature is enabled on a Content Engine. To reenable the Fast Cache feature on a standalone Content Engine, use the `wmt fast-cache enable` global configuration command.

**Step 2** Set the maximum delivery rate allowed per media player when Fast Cache is used to serve packets to the player by entering the following command:

```
ContentEngine(config)# wmt fast-cache max-delivery-rate number
```

**Note** The default maximum Fast Cache speed on the Content Engine is 5 (which can be changed with the `wmt fast-cache max-delivery-rate` global configuration command). To configure the maximum bit rate for the Fast Cache feature, use the `wmt fast-cache max-delivery rate` global configuration command to configure the Fast Cache speed multiplier. The value can be from 1 to 65535 (a value of 1 is equivalent to the Fast Cache feature being disabled). The maximum value for the bit rate is associated with the WMT license on the Content Engine.

When Fast Cache is configured, the Content Engine serves content using either the bit rate specified in the client request or the maximum delivery rate configured for Fast Cache in the Content Engine, whichever is smaller.

**Step 3** Verify that Fast Cache is enabled by entering the `show wmt` EXEC command.

---

### Configuring Transparent Redirection of WMT Requests

This section describes how to enable and configure a router and a standalone Content Engine for transparent redirection of RTSP requests:

#### Configuring RTSP Transparent Redirection of WMT Requests

With transparent redirection of WMT requests, a WCCP Version 2-enabled router or a Layer 4 switch transparently redirects WMT RTSP requests to the Content Engine (acting as a transparent proxy server). WMT RTSP transparent redirection is used to support WMT transparent caching on a standalone Content Engine that is running the ACNS 5.3.1 software and later releases. With this type of transparent redirection, you must configure WMT RTSP redirection on the WCCP Version 2-enabled routers or the Layer 4 switch as well as on the standalone Content Engine that will receive these redirected WMT MMS requests.

To configure WMT transparent redirection of WMT requests (WMT RTSP redirection) through WCCP Version 2, you must perform both of these tasks:

- Configure WMT RTSP transparent redirection (WCCP Version 2 services 80 and 83) on the WCCP Version 2 routers that will support this Windows Media service
- Configure WMT RTSP transparent redirection on the standalone Content Engine
The following example shows how to use the Content Engine CLI to configure WMT RTSP transparent redirection through WCCP Version 2. This example assumes that you have enabled the licensed WMT feature on the standalone Content Engine, as described in the “Enabling WMT Licenses on Standalone Content Engines” section on page 9-17.

To configure WMT RTSP transparent redirection through WCCP Version 2, follow these steps:

**Step 1** Enable WCCP Version 2 on the router (Router A).

```
RouterA# configure terminal
RouterA(config)# ip wccp version 2
```

**Step 2** Enable the WCCP Version 2 services 80 and 83 on the router.

- **a.** Enable service 80 (the rtsp redirection service) on Router A.
  
  ```
  RouterA(config)# ip wccp 80
  ```

- **b.** Enable service 83 (the wmt-rtspu redirection service) on Router A.
  
  ```
  RouterA(config)# ip wccp 83
  ```

**Note** To perform WCCP transparent redirection of WMT RTSP traffic, you must enable service 80 and service 83 on the WCCP Version 2-enabled router.

**Step 3** Specify an interface on which the RTSP redirection services will run on Router A.

```
RouterA(config)# interface type number
```

The following shows how to configure the outgoing interface to the Internet as Ethernet 0 on Router A:

```
RouterA(config)# interface Ethernet 0
```

**Step 4** From interface configuration mode on Router A, enable WCCP redirection to service 80 and 83 on the specified router interface (in this case, the outgoing interface).

Specify the inbound or outbound interface for service 80 and service 83.

```
RouterA(config-if)# ip wccp 80 redirect out
RouterA(config-if)# ip wccp 83 redirect out
```

**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 (using the `ip wccp service number redirect in` interface configuration command). This depends primarily on your network topology.

**Step 5** Enable WMT RTSP redirection through WCCP on the standalone Content Engine that will be functioning as the transparent proxy server for redirected WMT requests from Router A:

- **a.** Enable WCCP Version 2 on the Content Engine.
  
  ```
  ContentEngine(config)# wccp version 2
  ```

- **b.** Create the numbered router list that you want to associate with service 80 and 83.

  In the following example, there is one WCCP Version 2-enabled router (Router A) associated with router lists 1. Router A has an IP addresses of 172.16.25.25:
Configuration Transparent Redirection of WMT Requests

ContentEngine(config)# wccp router-list 1 172.16.25.25

**c.** Enable the router list (router list 1) that you just created in Step b.

ContentEngine(config)# wccp wmt router-list-num 1

**Step 6** Enable transaction logging on the standalone Content Engine.

ContentEngine(config)# transaction-log enable

**Tip**
You can configure standalone Content Engines to log usernames for any authenticated WMT RTSP requests. For more information, see the “Enabling the Logging of Usernames to the WMT Transaction Log” section on page 9-54.

**Step 7** Save the new configuration on the Content Engine.

ContentEngine# copy running-config startup-config

**Step 8** Verify that WMT is now running on the Content Engine.

ContentEngine# show wmt

**Step 9** Configure WMT parameters (for example, configure the WMT bandwidth) as needed using CLI commands or the Content Engine GUI.

**Step 10** After starting the Windows Media 9 player, display all of the WMT statistics for this Content Engine:

ContentEngine# show statistics wmt all

Objects transported over HTTP are counted in the HTTP statistics, and not included in the `show statistics wmt all` command output. In the ACNS 5.3.1 software and later releases, the command output also includes WMT statistics for objects transported over RTSP if WMT RTSP transparent redirection has also been configured (services 80 and 83).

After configuring the routers and Content Engine to support WMT RTSP transparent redirection through WCCP Version 2, enable and configure WMT transparent caching on the Content Engine, as described in the “Enabling and Configuring WMT Transparent Caching on Standalone Content Engines” section on page 9-35.

To enable the Content Engine to receive RTSP requests that are transparently redirected to it by a Layer 4 switch, enter the following command:

ContentEngine(config)# rtsp l4-switch enable

The `rtsp l4-switch enable` global configuration command allows the Content Engine to receive RTSP requests that are transparently redirected to it by a Layer 4 switch (for example, a CSS switch). The Layer 4 switch intercepts the RTSP request from the client and transparently redirects that request to the Content Engine.
Enabling and Configuring WMT Caching on Standalone Content Engines

This section describes how to enable and configure the following types of WMT caching on standalone Content Engines:

- **Enabling and Configuring Nontransparent WMT Proxy Caching on Standalone Content Engines**, page 9-34
- **Enabling and Configuring WMT Transparent Caching on Standalone Content Engines**, page 9-35

Enabling and Configuring Nontransparent WMT Proxy Caching on Standalone Content Engines

With direct proxy routing, the client WMT media players send their requests directly to the Content Engine that is acting as a nontransparent forward proxy server. Direct proxy routing is used to support WMT proxy caching on a Content Engine. With direct proxy routing, you must point the client WMT media players directly to the Content Engine.

For more information about pointing the client WMT media players directly to the Content Engine, see the following sections:

- **Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests**, page 4-43
- **Pointing Windows Media Players Directly to a Standalone Content Engine for WMT MMS Requests**, page 4-45

To use the Content Engine CLI to enable and configure WMT proxy caching on a standalone Content Engine, follow these steps:

**Step 1** Specify the maximum size of a single object that the Content Engine should store in its WMT cache. The range of values is between 1 and 1,000,000 megabytes. The default value is 1024 megabytes.

```
ContentEngine(config)# wmt cache max-obj-size size
```

**Step 2** Specify the external WMT server that the Content Engine is to use as an upstream WMT server (the outgoing HTTP proxy server for WMT) with the `wmt proxy outgoing` global configuration command. For more information, see the “Configuring Standalone Content Engines to Distribute VOD Files” section on page 9-35.

**Step 3** Enable WMT caching on the standalone Content Engine if it is not already enabled.

```
ContentEngine(config)# wmt cache enable
```
Enabling and Configuring WMT Transparent Caching on Standalone Content Engines

With WCCP routing or Layer 4 switching, you must configure the WCCP Version 2-enabled routers or Layer 4 switches and the Content Engine (transparent proxy server) for WMT RTSP transparent redirection to accept redirected WMT requests from Windows Media 9 players. For more information, see the “Configuring Transparent Redirection of WMT Requests” section on page 9-31.

To use the Content Engine CLI to enable and configure WMT transparent caching on a standalone Content Engine, follow these steps:

Step 1 Enable WMT caching on the standalone Content Engine if it is not already enabled.

```
ContentEngine(config)# wmt cache enable
```

Step 2 Specify the maximum size of a single object that the Content Engine should store in its WMT cache. The range of values is between 1 and 1,000,000 megabytes. The default value is 1024 megabytes.

```
ContentEngine(config)# wmt cache max-obj-size size
```

Step 3 Specify the list of routers from which this Content Engine will accept redirected WMT requests.

```
ContentEngine(config)# wccp wmt router-list number
```

Step 4 If you have not yet created a list of routers from which you want this Content Engine to accept redirected WMT requests, then create the router list now:

```
ContentEngine(config)# wccp router-list number
```

In the following example, there are two WCCP Version 2-enabled routers associated with router list 1. These routers have the IP addresses 172.16.25.25 and 172.16.26.26.

```
ContentEngine(config)# wccp router-list 1 172.16.25.25 172.16.26.26
```

Step 5 If you have not yet enabled the router list (for example, router list 1) that includes the WCCP Version 2-enabled routers that will redirect WMT requests to this Content Engine, then enable it.

```
ContentEngine(config)# wccp wmt router-list-num number
```

Step 6 Enable WCCP Version 2 on the Content Engine, if it is not already enabled.

```
ContentEngine(config)# wccp version 2
```

Step 7 Specify the external WMT server that the Content Engine should use as its upstream WMT server by using the `wmt proxy outgoing` global command.

Configuring Standalone Content Engines to Distribute VOD Files

You can preload VOD files on the Content Engine for on-demand delivery of these files to Windows Media clients. VOD caching is similar to HTTP caching; however, VOD files are cached in a different file system (mediafs) on the standalone Content Engine. WMT transparent caches and WMT proxy caches both support VOD caching.
To configure a standalone Content Engine to distribute VOD files to Windows Media clients, follow these steps:

**Step 1** Preload the VOD files on this Content Engine.

- a. Enable content preloading on the Content Engine.
- b. Use a preload URL list file to indicate which Windows Media content is to be preloaded on the Content Engine.
- c. Configure bandwidth control for preloading.
- d. Schedule or force an immediate preloading of the content.

**Step 2** Publish the URLs of the preloaded VOD files that clients can now access through their Windows Media players.

For instructions on how to preload files on a standalone Content Engine, see the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2. For information about how you can verify that the preloaded VOD files are being cached and properly distributed to clients, see the next section, “Verifying That Preloaded VOD Files Are Cached and Properly Distributed to Windows Media Clients.”

**Verifying That Preloaded VOD Files Are Cached and Properly Distributed to Windows Media Clients**

This section describes how you can verify that a standalone Content Engine has stored the preloaded VOD files in its cache, and is distributing these files to Windows Media clients upon request. This example assumes the following:

- Preloading has been configured on the Content Engine, the preload URL list includes some Windows Media files, and the Content Engine has completed the preload operation. For more information on this topic, see the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2.
  - A Windows Media player on at least one of your client desktops has been configured to point directly to the Content Engine (the nontransparent forward proxy server for this WMT client). For information on this topic, see the “Pointing Windows Media 9 Players Directly to a Standalone Content Engine for WMT RTSP Requests, page 4-43”.

To point to a VOD source and verify that both WMT proxy caching and WMT transparent caching are working properly on the standalone Content Engine when the clients are Windows Media 6 or 7 players, follow these steps:

**Step 1** Launch the Windows Media 6 or 7 player from one of your client’s personal computers (Client A) that is either configured to point directly to the Content Engine (nontransparent forward proxy server) or configured not to point directly to the Content Engine.

**Step 2** From the Windows Media player, choose **File > Open URL**.

**Step 3** Enter a URL that points to a Windows Media streaming file (for example, a *.asf or *.wmv file) that has been preloaded on the Content Engine.

The specified preloaded video should start playing in the Windows Media player on the client’s desktop.
Chapter 9      Configuring WMT Streaming Media Services on Standalone Content Engines

Configuring Standalone Content Engines to Deliver WMT Live Streams

Based on the capabilities and limitations of the network, standalone Content Engines can receive live WMT streams and then deliver WMT streaming content through multicast out or unicast out.

This section describes how to configure standalone Content Engines to deliver WMT live streams, and includes the following sections:

- Configuring Standalone Content Engines to Multicast Live WMT Streams, page 9-37
- Configuring an Alternative Source URL (Source Failover) for a WMT Multicast, page 9-42

Configuring Standalone Content Engines to Multicast Live WMT Streams

You can configure standalone Content Engines to send live content to Windows Media clients through multicasting or unicast. This section describes how to configure standalone Content Engines to relay live content through multicasting:

- Configuring Multicast-In Multicast-Out on Standalone Content Engines, page 9-38
- Configuring Unicast-In Multicast-Out on Standalone Content Engines, page 9-39
- Defining WMT Multicast Stations and Multicast Schedules on Standalone Content Engines, page 9-40
- Starting and Stopping WMT Multicast Stations, page 9-41

Note

You must enable WMT on the Content Engine before you can use the wmt multicast and wmt broadcast global configuration commands. See the “Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines” section on page 9-14.
To enable WMT multicasting for unicast-in multicast-out ("Configuring Unicast-In Multicast-Out on Standalone Content Engines" section on page 9-39) and multicast-in multicast-out ("Configuring Multicast-In Multicast-Out on Standalone Content Engines" section on page 9-38) on a standalone Content Engine, use the following command:

```
wmt multicast {schedule-start name minute hour day month | station-configuration name dest_addr dest_port media_source [play-forever]}
```

To configure the Time To Live (TTL) for a WMT multicast, use the `wmt multicast time-to-live ttl` global configuration command. The TTL value is specified as the number of hops. The value can be from 0 to 255 hops. The default is five hops. For example:

```
ContentEngine(config)# wmt multicast time-to-live 10
```

For information about how to configure standalone Content Engines to relay live content to clients through unicast, see the “Configuring an Alternative Source URL (Source Failover) for a WMT Multicast” section on page 9-42.

### Configuring Multicast-In Multicast-Out on Standalone Content Engines

The multicast-in multicast-out multicast receive feature allows you to receive multicast WMT streams delivered through IP multicasting, and then send 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 from multicast to unicast (and vice versa).

In this multicasting situation, a description file *.nsc is created that is accessible through multicast-out to clients. This is similar to the unicast-in multicast-out situation except that the input source is multicast. The clients use this description file to subscribe to the multicast.

To configure a standalone Content Engines to use multicast-in multicast-out to relay live WMT streams to Windows Media clients, follow these steps:

---

**Step 1** Configure a multicast station on the Content Engine by using the `wmt multicast station-configuration` global configuration command:

The syntax of this command is as follows:

- `station-configuration` configures the WMT multicast station on the Content Engine.
- `name` specifies the name of the WMT multicast station.
- `dest_addr` is the destination IP address (multicast IP address) of the WMT multicast station.
- `dest_port` is the destination port (1–65535) of the WMT multicast station.
- `media_source` is the media source of the multicast.

In the following example, a multicast station named acme is configured and used by the Content Engine as the multicast source file. Its Class D multicast IP address is 233.33.33.34, and the multicast port is 6667. The multicast stream stops playing once the end of the source.nsc file is reached, unless the `play-forever` option is specified.

```
ContentEngine(config)# wmt multicast station-configuration acme 233.33.33.34
6667 http://172.16.30.31/source.nsc
```

Note that in the ACNS 5.3 and later releases, the source TCP port on the multicast must be unique from any other multicast WMT station configured. This applies even if a different source server is used. The source TCP ports must be unique across all multicast stations.
In the ACNS 5.3.1 software and later releases, rtsp://, rtspu://, and rtsp:// URLs for WMS 9 (that is if the client is a Windows Media 9 player and the server is the Windows Media 9 server running on the Content Engine) as well as http:// URLs are supported.

**Step 2** Start the multicast.

```
ContentEngine# wmt multicast-station start acme
ContentEngine#
```

**Step 3** Open your WMT media player and choose **File > Open URL**.

**Step 4** Enter the following URL:

```
http://ContentEngineIPaddress/acme.nsc
```

**Step 5** Click **OK**.

The Windows Media client should receive the media file specified in **Step 1**.

---

### Configuring Unicast-In Multicast-Out on Standalone Content Engines

The Content Engine supports several different sources for a unicast-in multicast-out stream, otherwise known as stream splitting. A 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.

**Tip**

If a live stream is interrupted on the server side, you must stop the multicast station and then restart the same station to resume live multicasting. Use the `wmt multicast-station stop name` EXEC command to stop this station. Use the `wmt multicast-station start name` EXEC command to restart the same station.

The unicast-in multicast-out multicast delivery feature allows 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.
To configure a standalone Content Engines to use unicast-in multicast-out to send live WMT streams to WMT clients, follow these steps:

**Step 1** Configure a multicast station on the Content Engine by using the `wmt multicast station-configuration` global configuration command.

In the following 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 239.33.33.33, and the multicast port is 3333. The `play-forever` option is used. When the input source.asf file is a VOD file, this option automatically restarts playback of the file from the beginning of the source.asf file once the end of this file has been reached. This source file source.asf can be located on any Windows WMT server.

```
ContentEngine(config)# wmt multicast station-configuration test1 239.33.33.33 3333 rtp://172.16.30.31/source.asf play-forever
```

**Step 2** Start the multicast.

```
ContentEngine# wmt multicast-station start test1
ContentEngine#
```

**Step 3** Open your WMT media player and choose *File > Open URL*.

**Step 4** Enter the following URL:

```
http://ContentEngineIPaddress/test1.nsc
```

**Step 5** Click *OK*.

The Windows Media player should retrieve the multicast description .nsc file and join the multicast station that is specified in Step 1.

The use of port 80 is implied in the URL for WMT multicasting, such as the following:

```
http://ContentEngineIPaddress:80/test1.nsc
```

---

**Defining WMT Multicast Stations and Multicast Schedules on Standalone Content Engines**

To configure a WMT multicast station on a standalone Content Engine, use the `wmt multicast station-configuration` global configuration command.

A multicast station is a defined location (a multicast IP address and multicast port) from which a player can receive streams. This multicast IP address is not related to the IP address of the Content Engine.

The `wmt multicast station-configuration name dest_addr dest_port media_source` command specifies a multicast station name, a multicast IP address, port number, and media source for the multicast station created. Each WMT multicast 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
Chapter 9 Configuring WMT Streaming Media Services on Standalone Content Engines

Configuring Standalone Content Engines to Deliver WMT Live Streams

You must choose a multicast IP address that does not conflict internally within the same multicast-enabled network configuration. For a complete table of unusable multicast address ranges, see Table B-8 in the “Unusable Multicast Address Assignments” section on page B-11.

The destination port of the WMT multicast station is specified by the `dest_port` option. Valid options are 1 through 65535. However, the multicast-enabled network may impose certain restrictions on your choice of port. Normally, port numbers less than 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. The `play-forever` option configures the stream to loop and restart. The default is to play the stream once and stop.

For example:

```
ContentEngine(config)# wmt multicast station-configuration acme 239.33.33.33 3333 rtsp://172.16.30.31/source.asf play-forever
```

In this example:
- The name of the WMT multicast station is `acme`.
- The multicast IP address of the WMT multicast station is `239.33.33.33`.
- The destination port of the WMT multicast station is `3333`.
- The source of the multicast is `rtsp://172.16.30.31/source.asf`, and it will play forever.

To configure multicasting schedules for WMT multicast stations, use the `wmt multicast station-configuration` global configuration command.

The `schedule-start name minute` option creates a scheduling option to allow the Content Engine to start a multicast at a specified time.

Table 9-10 describes the command parameters for the `wmt multicast station-configuration schedule-start` global configuration command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>station-name</td>
<td>Name of the WMT multicast station for which you are creating the schedule.</td>
</tr>
<tr>
<td>schedule-start</td>
<td>Configures an automatic start schedule.</td>
</tr>
<tr>
<td>minute</td>
<td>Start time minute (0–59).</td>
</tr>
<tr>
<td>now</td>
<td>Start the WMT multicast station now. If you specify this option, the multicast station will be started immediately. If the station is running and the Content Engine is reloaded, the station will be automatically started again after the reload.</td>
</tr>
</tbody>
</table>

Starting and Stopping WMT Multicast Stations

In the ACNS 5.3.1 software and later releases, the ability to resume a WMT multicast automatically after a Content Engine is reloaded is supported. To support this feature, the `wmt multicast station-configuration station-name schedule-start now` global configuration command was added in the ACNS 5.3.1 software release. If you specify this command for a specific multicast station, the
multicast station will be started immediately and then automatically started again after the
Content Engine is reloaded. The reason for introducing this new global configuration command is that
the wmt multicast-station start station-name EXEC command is not persistent across reboots (that is,
if a multicast station is running before the Content Engine is reloaded, it will not continue to run after
the reload).

The no wmt multicast station-configuration station-name schedule-start now global configuration
command, which was added in the ACNS 5.3.1 software release, works exactly like the wmt
multicast-station stop station-name EXEC command. You can use either command to stop a specific
WMT multicast station.

The wmt multicast-station start EXEC command only works if you have configured a multicast station
first, using the wmt multicast station-configuration station-name global configuration command.

For example, after using the wmt multicast station-configuration station-name global configuration
command to configure a multicast station, you can start or stop the multicast station by using the
wmt multicast-station EXEC command:

wmt multicast-station { start station-name | stop station-name }

Table 9-11 describes the command parameters for the wmt multicast-station EXEC command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>multicast-station</td>
<td>Sets the WMT multicast stations to start or stop.</td>
</tr>
</tbody>
</table>
| start             | Starts a WMT multicast station. If you use this option to start a multicast
                  station and it is running before the Content engine is reloaded, the station
                  will not continue to run after the Content engine is reloaded.          |
| station-name      | Name of the WMT multicast station to be started.                            |
| stop              | Stops a WMT multicast station.                                              |
| station-name      | Name of the WMT multicast station to be stopped.                            |

The following examples demonstrate the start and stop options on the multicast station named acme.

ContentEngine# wmt multicast-station start acme
ContentEngine# wmt multicast-station stop acme

Configuring an Alternative Source URL (Source Failover) for a WMT Multicast

In the ACNS 5.3.1 software and later releases, you can configure an alternative source URL (source
failover) for a WMT multicast with the wmt multicast station-configuration station-name failover
global configuration command:

ContentEngine(config)# wmt multicast station-configuration acme failover ?
  alternate-source Alternate source url
  retry-count     No. of retries for all sources
  retry-interval  Sleep interval between retries

Use the alternate-source option to specify a fully-qualified alternative source URL for the WMT
multicast.

To specify the number of retries for all of the sources, use the retry-count option. The retry count can
be from 0 to 2000. To specify the retry interval (that is, the amount of time that the Content Engine is to
wait between retries), use the retry-interval option. The retry interval can be from 0 to 60 minutes.
Configuring Standalone Content Engines to Deliver WMT Live Streams

You can configure standalone Content Engines to send live content to WMT clients through multicasting or unicast. This section describes how to configure standalone Content Engines to relay live content through unicast:

- Configuring Multicast-In Unicast-Out on Standalone Content Engines, page 9-43
- Configuring Unicast-In Unicast-Out on Standalone Content Engines, page 9-44

These sections also describe how to configure a WMT broadcast alias on a standalone Content Engine for unicast-out.

Configuring Multicast-In Unicast-Out on Standalone Content Engines

The multicast-in unicast-out feature allows 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. To configure a multicast-in unicast-out broadcast on a standalone Content Engine, use the \texttt{wmt broadcast \{alias-name name source url\}} global configuration command. 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.

In this situation, a unicast-out publishing point is created to deliver the incoming stream live to requesting clients.

To configure a standalone Content Engines to use multicast-in unicast-out (unicasting out) to relay live WMT streams to WMT clients, follow these steps:

**Step 1** Configure a WMT broadcast alias on the Content Engine:

```
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. The source of an alias in the format http://server/file.nsc signals the Content Engine to treat this source as a multicast input source.

**Step 2** Open your WMT media player and choose \texttt{File > Open URL}.

**Step 3** Enter the following URL:

```
rtsp://ContentEngineIPaddress/myunicast
```

**Step 4** Click \texttt{OK}.

The WMT media player should receive the media source file specified in Step 1. In this situation, an RTSP URL is used to access the streaming media, and only the alias name is specified instead of the *.nsc files as in the multicast-out situations.

This converts the multicast stream to unicast and sends it to the requester (the WMT client).
Configuring Unicast-In Unicast-Out on Standalone Content Engines

The unicast-in unicast-out feature provides a point-to-point connection between the client and the Content Engine. The advantage of unicasting when streaming media over a network is that only a single stream needs to be pulled over the network between the origin server and Content Engine, but that stream can be delivered to multiple clients in a nonmulticast environment. A server running Windows Media Services can provide a unicast video stream to multiple clients through a single stream delivered to the Content Engine. Unicast-in unicast-out is typically used to broadcast live events.

In this situation, unicast-in unicast-out provides a point-to-point connection between the client and the Content Engine. The Content Engine in turn makes a single connection to the media server. Multiple requests for the same stream can be split by the Content Engine so that each client receives a distinct data stream directly from the Content Engine, while the Content Engine maintains its single stream connection to the media server.

You can configure unicast-in unicast-out in the following ways:

- By live splitting without any configuration.
  
  In this case, the Content Engine acts as a proxy. When clients request the same unicast URL, the Content Engine proxy automatically splits the stream from the source to the clients.

- By configuring the Content Engine with a broadcast alias.
  
  In this case, a client makes the request to the Content Engine as if it were the Windows Media Server, and the Content Engine checks to see whether the incoming stream is present. If it is, then the Content Engine joins the stream and splits it to the new client. If the request is the first client request for this stream, the Content Engine sends the request out to the server and then serves it to the new client.

To configure a standalone Content Engine to use unicast-in unicast-out (unicast out) to relay live WMT streams to clients, follow these steps:

---

**Step 1** From the Content Engine GUI, choose **Caching > WMT-Streaming**. The WMT Streaming window appears.

**Step 2** Click **WMT Config**. The WMT Configurations window opens.

**Step 3** Click the **Broadcast Unicast Publishing** link. The WMT Broadcast Unicast Publishing window appears.

**Step 4** In the Alias Name field, enter a broadcast alias for the live broadcast configuration (for example, broadcast1).

**Step 5** In the Source field, enter the broadcast source for the live broadcast configuration using the following format:

\(<\text{protocol}>://\text{server-name:port-num/path/file-name}\)
The variables are as follows:

- **protocol** is either HTTP or RTSP.
- **server-name** is the name of the server.
- **port-num** is the port number. The default is port 8080 for HTTP and port 554 for RTSP.
- **path** is the full pathname.
- **file-name** is a media filename if the file is in the content root directory.

For example:

```
rtsp://wms.company.com/cotv
```

wms.company.com is the name of the Windows Media Server, and cotv is the name used when the broadcast alias is created.

**Step 6** Click **Update** to save the settings.

**Step 7** Open your WMT player and choose **File > Open URL**. Enter the following URL:

```
rtsp://ContentEngineIPaddress/broadcast1
```

- **ContentEngineIP address** is the IP address or domain name of the Content Engine.
- **broadcast1** is the broadcast alias specified in **Step 4**.

**Step 8** Click **OK**.

The WMT player should receive the media source file specified in **Step 5**. In this situation, an RTSP URL is used to access the streaming media, and only the broadcast alias (for example, broadcast1) is specified instead of the *.nsc files in the multicast-out situations. This converts the multicast stream to unicast and sends it to the WMT client.

---

**Clearing WMT Streams on Standalone Content Engines**

To clear WMT streams on a standalone Content Engine, use the **clear wmt** EXEC commands:

```
ContentEngine# clear wmt ?
 incoming  Clear all incoming WMT streams
 outgoing   Clear all outgoing WMT streams
 stream-id  Stream Id of the WMT stream to be cleared
```

**Table 9-12** describes these **clear wmt** EXEC commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear wmt incoming</td>
<td>Clears all incoming WMT streams from the Content Engine. Also stops all of the Content Engine’s WMT processes that are associated with incoming WMT streams.</td>
</tr>
</tbody>
</table>
Displaying Information about the WMT RTSP Server Configuration

In the ACNS 5.3.1 software and later releases, by entering the `show rtsp server wmt` EXEC command, you can display information about the WMT RTSP server that runs on the Content Engine. The following sample output shows a sample configuration for the WMT RTSP server that is running on the standalone Content Engine:

```
Content Engine# show rtsp server wmt
WMT version: ce507-001.000
WMT license key is installed
WMT evaluation is not enabled
WMT end user license agreement accepted
WMT is enabled
WMT disallowed client protocols: none
WMT outgoing bandwidth configured is 1 Kbits/sec
WMT incoming bandwidth configured is 56000 Kbits/sec
WMT max sessions configured: 2500
WMT max sessions platform limit: 2500
WMT max sessions enforced: 2500 sessions
WMT max outgoing bit rate allowed per stream: 2 Kbits/sec
WMT max incoming bit rate allowed per stream: 3 Kbits/sec
WMT debug level: 0
WMT L4 switch is enabled
WMT debug client ip not set
WMT debug server ip not set
WMT fast-start is enabled
WMT fast-start max. bandwidth per player is 65 (Kbps)
WMT fast-cache is enabled
WMT fast-cache acceleration factor is 5
WMT Extended Transaction Log is not enabled
WMT Transaction Log format is Windows Media Services 4.1 logging
```

Displaying Information about the Current WMT Configuration

To display the current WMT configuration for a standalone Content Engine, enter the `show wmt` EXEC command. To display the current WMT configuration for a standalone Content Engine, enter the `show wmt` EXEC command.
Displaying WMT Statistics

To display statistics about WMT requests, use the `show statistics wmt` EXEC commands.

```
ContentEngine# 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
  rule       Display rule template statistics
  savings    Display unicast savings statistics
  streamstat Display Windows Media streaming connections
  urlfilter  Display urlfiltering statistics for mms and rtsp requests
  usage      Display concurrent usage statistics
```

ContentEngine#

In the ACNS 5.3.1 software and later releases, the output of the `show statistics wmt` EXEC commands includes information about WMT RTSP requests. For example, the output from the `show statistics wmt` EXEC commands was changed as follows:

- RTSP-related information was added to the `show statistics wmt all` command output.
- Information about RTSPT and RTSPU were added in the transport protocol portion of the `show statistics wmt bytes` command output.
- RTSPT and RTSPU errors were added to the `show statistics wmt errors` command output.
- The `show statistics wmt requests` command output includes the RTSPT and RTSPU protocols as well as Fast Start and Fast Cache data.

In the ACNS 5.3.1 software and later releases, you can display aggregated live statistics by entering the `show statistics wmt streamstat live` EXEC command.

Using WMT Logging with Standalone Content Engines

This section describes how to use the WMT logging features, and covers the following topics:

- Using WMT Multicast Logging, page 9-48
- Using WMT Transaction Logging, page 9-48
- Using WMT Error Logging, page 9-55
Using WMT Multicast Logging

WMT logs are logged to a working log on the local disk in one of the following files, depending upon where the sysfs is mounted on the Content Engine:

- The file named /local1/logs/export/working.log
- The file named /local2/logs/export/working.log

To provide a log of multicast statistics to multicast server administrators, use the log option of the wmt multicast station-configuration global configuration command:

```
wmt multicast {station-configuration name dest_addr dest_port media_source
[log {local | webserver webserver_url}]})
```

- To enable logging of multicast URLs, specify the log option.
- To configure logging of multicast URLs to a local disk, specify the local option.
- To configure logging of multicast URLs to a web server, specify the webserver option and enter the URL to identify the location of the web server.

The variables for this command are as follows:

- name is the name of the WMT multicast station.
- dest_addr is the WMT multicast station destination IP address.
- dest_port is the WMT multicast station destination port (1–65535).
- media_source is the WMT multicast media source (for example, http://live/live).
- webserver_url specifies the fully qualified webserver URL.

These statistics include the multicast IP address, port number, start time, and number of clients. When configuring this option, you have the choice to provide either a local URL where the multicast logging statistics can be sent, or an external fully qualified server URL that can receive these statistics. The multicast logging URL option can point to the multicast server itself or to any web server that is capable of processing the posted information from the users who subscribed to the multicast address.

The following example displays the multicast logging statistics sent to the multicast server:

```
{5DC90EEB-CEB1-467C-9F7A-BCF5EDEE3FF} 10.1.0.3055 en-US - -
wmplayer.exe 10.1.0.3055
Windows_2000 10.0.0.2195 Pentium 0 152543 65389
asfm UDP WINDOWS_MEDIA_AUDIO_V2
MICROSOFT_MPEG-4_VIDEO_CODEC_V3 http://172.16.192.91/cisco.nsc
- 166245 - 176 0 0 0 0 0 1
0 100 233.0.4.5 -- --
```

Using WMT Transaction Logging

For some companies, streaming media is a source of revenue, and therefore needs to be tracked closely. Because these companies charge their customers to stream on-demand content and live broadcasts, they must rely on logged information to track what content a particular customer viewed, how long they viewed it, and the viewing quality. Consequently, the accuracy and reliability of transaction logging is very important to these companies.

The Windows Media Services 9 Series provides a more robust logging model than Windows Media Services Version 4.1. In the ACNS 5.2.1 software, support for Windows Media Services 9 logging was added.
In the ACNS 5.2.1 software and later releases, the following logging formats are supported for WMT transaction logging:

- Standard Windows Media Services 4.1
- Extended Windows Media Services 4.1
- Standard Windows Media Services 9.0
- Extended Windows Media Services 9.0

**Note**

In the ACNS 5.1 software and earlier releases, only the standard Windows Media Services 4.1 and the extended Windows Media Services 4.1 logging formats were supported.

The extended versions of the logging formats are extensions to the standard logging format and contain additional fields that are Content-Engine specific (for example, the CE-action field that specifies whether it was a cache hit or miss, and the CE-bytes field that specifies the number of bytes that were sent out from the Content Engine.

The Content Engine’s transaction logging format for WMT streaming is consistent with that of the Windows Media Services and the World Wide Web Consortium (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. When transaction logging is enabled, daemons create a separate working.log file in /local1/logs/export for WMT transactions.

All client information in the transaction logs is sent to the origin server by default.

Windows Media players connect to a Windows Media server using the following protocols:

- Windows Media players earlier than Version 9.0 (Windows Media 6 and 7 players) use HTTP 1.0 or the MMS protocol.
- Windows Media 9 players use HTTP 1.0, HTTP 1.1, and RTSP.

Depending on the version of the Windows Media player, logs are sent in different formats, such as text, binary, or XML. See Table 9-13.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Player and Distributor</th>
<th>Log Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/1.0</td>
<td>Windows Media players earlier than Version 9.0 (for example, Windows Media 6.4 or 7.0 players) Content Engine (caching and proxy server) is running Windows Media Services Version 9.0 and streaming from a WMT server that is running Windows Media Services 4.1</td>
<td>World Wide Web Consortium (W3C) standard space-delimited text log</td>
</tr>
</tbody>
</table>
In the ACNS 5.2.1 software and later releases, XML logging for MMS-over-HTTP is supported. The posted XML log file from the Windows Media player to the Content Engine (Windows Media server) can be parsed and saved to the normal WMT transaction logs that are stored on the Content Engine.

\[\text{Note}\]

In the ACNS 5.3.1 software and later releases, support for WMS 9 logging for WMT RTSP requests from Windows Media 9 players is available.

### Specifying the Format of the WMT Transaction Logs

To specify the format for the WMT transaction logs on standalone Content Engines, use the `wmt transaction-logs format` global configuration command that is supported in the ACNS 5.2.1 software and later releases.

```
wmt transaction-logs format { extended { wms-41 | wms-90 } | wms-41 | wms-90 }
```

By default, the standard Windows Media Services 4.1 logging format is used (no Content Engine-specific details are logged).

Table 9-14 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction-logs</td>
<td>Configures the logging format of the WMT transaction logs.</td>
</tr>
<tr>
<td>format</td>
<td>Sets the format for WMT transaction logs.</td>
</tr>
<tr>
<td>extended</td>
<td>Specifies the WMT extended format for transaction logs. Enables username logging in the WMT transaction log.</td>
</tr>
</tbody>
</table>
To log the username to the WMT transaction log, you must enable the extended WMT logging feature on the Content Engine, use the `wmt extended transaction-log enable` global configuration command. For more information, see the section, "Enabling the Logging of Usernames to the WMT Transaction Log" section on page 9-54.”
Extended Windows Media Services 9.0 Logging Format

ACNS 5.5 software forwards two types of logs to the Windows Media Server:

- Logs sent by the Content Engine for its own connection to the Window Media Server.
- Logs forwarded from the Windows Media Player (client) for its playback. These logs are of two types: combination logs and render logs.

For more information, refer to the Windows Media Service 9.0 Logging Model documentation at the following URL:

http://www.microsoft.com/windows/windowsmedia/howto/articles/LoggingModel.aspx

Extended Windows Media 9.0 transaction logs are displayed using the following format:

c-ip date time c-dns cs-uri-stem c-starttime x-duration c-rate
c-status c-playerid c-playerversion c-playerlanguage cs(User-Agent)
cs(Referer) c-hostexe c-hostexevers c-os
c-osversion c-cpu filelength filesize avgbandwidth protocol transport audiocodec
c-pkts-lost-client c-pkts-lost-net c-pkts-lost-cont-net
c-resendreqs c-pkts-recovered-ECC
c-pkts-recovered-resent c-buffercount c-totalbuffertime c-quality s-ip s-dns
s-totalclients s-cpu-util CE-action CE-bytes Username

Table 9-15 describes the fields shown in this example.

<table>
<thead>
<tr>
<th>Field</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>c-ip</td>
<td>IP address of the client computer. A client that is not connected properly provides a client proxy server IP address, not the client IP address.</td>
</tr>
<tr>
<td>date</td>
<td>Date (according to Greenwich mean time) when an entry is generated in the log file.</td>
</tr>
<tr>
<td>time</td>
<td>Time (according to Greenwich mean time) when an entry is generated in the log file.</td>
</tr>
<tr>
<td>c-dns</td>
<td>DNS name of the client computer.</td>
</tr>
<tr>
<td>cs-uri-stem</td>
<td>Name of the file that is playing, an .asf file for unicast and an .asx file for multicast.</td>
</tr>
<tr>
<td>c-starttime</td>
<td>Time stamp (in seconds) of the stream when an entry is generated in the log file.</td>
</tr>
<tr>
<td>x-duration</td>
<td>Length of time a client played content before a client event (fast forward [FF], rewind [REW], pause, stop, or jump to marker). A log entry is generated whenever one of these client events occurs.</td>
</tr>
<tr>
<td>c-rate</td>
<td>Mode of Windows Media player when the last command event was sent.</td>
</tr>
<tr>
<td></td>
<td>• 1 = Windows Media player was paused or stopped during a play, fast-forward, rewind, or marker jump operation.</td>
</tr>
<tr>
<td></td>
<td>• -5 = Windows Media player was rewound from a play, stop, or pause operation.</td>
</tr>
<tr>
<td></td>
<td>• 5 = Windows Media player was fast-forwarded from a play, stop, or pause operation.</td>
</tr>
<tr>
<td>c-status</td>
<td>Codes that describe client status. Mapped to HTTP/1.1 and RTSP client status codes described in Request for Comments (RFC) 2068 and RFC 2326. Windows Media Services includes the extensible client status codes 480 (simultaneous client connections exceeded the maximum client limit of the server) and 483 (stream exceeded maximum file bit rate limit of the server).</td>
</tr>
<tr>
<td>c-playerid</td>
<td>Globally unique identifier (GUID) of the player.</td>
</tr>
<tr>
<td>c-playerversion</td>
<td>Version number of the player.</td>
</tr>
</tbody>
</table>
### Table 9-15  Field Descriptions for Windows Media Services 9.0 Logs (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>c-playerlanguage</td>
<td>Language country code of the client computer.</td>
</tr>
<tr>
<td>cs(User-Agent)</td>
<td>Browser type used if Windows Media player was embedded in a browser.</td>
</tr>
<tr>
<td>cs(Referer)</td>
<td>URL of the web page in which Windows Media player was embedded (if it was embedded).</td>
</tr>
<tr>
<td>c-hostexe</td>
<td>Host application; for example, a web page in a browser (iexplore.exe), a Microsoft Visual Basic applet (vb.exe), or standalone Microsoft Windows Media player (mplayer2.exe).</td>
</tr>
<tr>
<td>c-hostexever</td>
<td>Host application version number.</td>
</tr>
<tr>
<td>c-os</td>
<td>Operating system of the client computer.</td>
</tr>
<tr>
<td>c-osversion</td>
<td>Operating system version number of the client computer.</td>
</tr>
<tr>
<td>c-cpu</td>
<td>CPU type of the client computer.</td>
</tr>
<tr>
<td>filesize</td>
<td>Size of the file (in bytes). This value is 0 for a live stream.</td>
</tr>
<tr>
<td>avgbandwidth</td>
<td>Average bandwidth (in bits per second) at which the client was connected to the server.</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol used to access the stream: HTTP or ASFM (multicast protocol).</td>
</tr>
<tr>
<td>transport</td>
<td>Transport protocol used to deliver the stream (UDP, TCP, or UDP over IP multicast).</td>
</tr>
<tr>
<td>audiocodec</td>
<td>Audio codec used in the stream.</td>
</tr>
<tr>
<td>videocodec</td>
<td>Video codec used to encode the stream.</td>
</tr>
<tr>
<td>channelURL</td>
<td>URL to the .nsc file. A unicast client information log file records a dash (-) for this field.</td>
</tr>
<tr>
<td>sc-bytes</td>
<td>Bytes sent by the server to the client.</td>
</tr>
<tr>
<td>c-bytes</td>
<td>Number of bytes received by the client from the server. For unicast, the c-bytes value and sc-bytes value must be identical. If not, packet loss has occurred.</td>
</tr>
<tr>
<td>s-pkts-sent</td>
<td>Total number of packets sent by the server.</td>
</tr>
<tr>
<td>c-pkts-received</td>
<td>Number of packets from the server (s-pkts-send) that are received correctly by the client on the first try.</td>
</tr>
<tr>
<td>c-pkts-lost-client</td>
<td>Number of packets lost during transmission from server to client and not recovered at the client layer through error correction or at the network layer through User Datagram Protocol (UDP) resends.</td>
</tr>
<tr>
<td>c-pkts-lost-net</td>
<td>Number of packets lost on the network layer.</td>
</tr>
<tr>
<td>c-pkts-lost-cont-net</td>
<td>Maximum number of continuously lost packets on the network layer during transmission from server to client.</td>
</tr>
<tr>
<td>c-resendreq</td>
<td>Number of client requests to receive new packets. This field contains a value only if the client is using UDP resend.</td>
</tr>
<tr>
<td>c-pkts-recovered-ECC</td>
<td>Number of packets repaired and recovered on the client layer. Packets repaired and recovered at the client layer are equal to the difference between c-pkts-lost-net and c-pkts-lost-client.</td>
</tr>
<tr>
<td>c-pkts-recovered-resent</td>
<td>Number of packets recovered because they were resent using UDP.</td>
</tr>
<tr>
<td>c-buffercount</td>
<td>Number of times the client buffered while playing the stream.</td>
</tr>
<tr>
<td>c-totalbuffertime</td>
<td>Time (in seconds) the client used to buffer the stream. If the client buffers the stream more than once before a log entry is generated, c-totalbuffertime is the total amount of time the client spent buffering the stream.</td>
</tr>
</tbody>
</table>
Enabling the Logging of Usernames to the WMT Transaction Log

If the Content Engine is configured to use the extended format of WMT transaction logging and the extended WMT logging feature is enabled, then the Content Engine logs usernames for any authenticated WMT requests. Usernames are logged not only for NTLM authentication but also for Negotiate, Digest, and basic authentication.

Negotiate and Digest authentication is applicable for the HTTP protocol only.

By default, the extended WMT logging feature is disabled. If the extended logging format is enabled (using the `wmt transaction-logs format extended` global configuration command) but the extended WMT logging feature is disabled, the username field in the WMT transaction log will be empty.

To enable the logging of usernames for any authenticated WMT request on standalone Content Engines, follow these steps:

**Step 1** Configure the Content Engine to use the extended Windows Media Services 4.1 or Windows Media Services 9 format for transaction logging by using the `wmt transaction-logs format extended` global configuration command.

For more information, see the “Specifying the Format of the WMT Transaction Logs” section on page 9-50.

**Step 2** Enable the Content Engine to log the usernames for any authenticated WMT request.

```
Content Engine(config)# wmt extended transaction-log enable
```
Windows Media Transaction Log Forwarding

Windows Media transaction logs are forwarded to a Windows Media Server or an upstream Content Engine only if log forwarding is enabled on both the Content Engine (by using the `wmt advanced server log-forwarding enable` global configuration command, which is enabled by default) and the Windows Media Server 9 (using the WMS Client logging plugin, which disabled by default). Log forwarding is supported for the RTSP protocol only.

To enable log forwarding on the Windows Media Server Version 9, follow these steps:

**Step 1** From the Windows Media Services Administration GUI, choose your Windows Media publishing point, and in the details pane on the right, click the Properties tab.

**Step 2** In the Category pane, choose **Logging**, and in the Plug-in pane, double-click **WMS Client Logging**. The WMS Client Logging Properties dialog box appears.

**Step 3** Choose the **Log Entries** tab and check the Sessions played from a player cache or a cache/proxy server check box.

**Step 4** Click **Apply**, and then click **OK**.

**Step 5** Disable the WMS Client Logging plugin, and then re-enabled it for log forwarding to take effect for that publishing point.

Using WMT Error Logging

In the ACNS 5.2.1 software release, WMT error logging was enhanced. More information is now logged about the following events:

- When a WMT client is abruptly disconnected
- When any WMT streams are cleared on the Content Engine

Error logs are in the same format and location as syslogs. The WMT log messages are logged to `/local1/errolog/wmt_errorlog.current`.

You can configure the Content Engine for WMT error logging by using the `debug wmt error` EXEC command. This command debugs WMT level 1 functionality.

```
ContentEngine# debug wmt error ?
   client-ip  Debug request from a specific client
   server-ip  Debug request to a specific server
```

- Specify the `client-ip cl-ip-address` option to debug the request from a specific client IP address to level 1 (show error).
- Specify the `server-ip sv-ip-address` option to debug the request from a specific server IP address to level 1 (show error).

There is also a `debug wmt trace` EXEC command that debugs WMT level 2 functionality (show error and trace). Content Engine performance is affected when you run the `debug wmt trace` command. Consequently, we recommend that the `debug wmt trace` command be used only at the direction of Cisco Systems technical support personnel.
Logging WMT Client Disconnects

When a WMT client is disconnected abruptly, the following information is logged in ACNS software error logs:

- Reasons for the client disconnect (for example, the request was blocked by the rules, the maximum incoming or outgoing bit rate limit was reached, the maximum incoming or outgoing bandwidth limit was reached).
- Client information (for example, client IP address, server IP address, the requested URL, client protocol, version of the client media player, the number of packets that the client received, and the number of packets that the server sent).

Logging the Clearing of WMT Streams on Standalone Content Engines

In the ACNS 5.2.1 software release, the error logs were enhanced to log a message when WMT streams are cleared and the associated processes are stopped on the Content Engine.

See Table 9-12 for a description of the Content Engine CLI commands, which you can use to clear WMT streams on a standalone Content Engine and which result in a message being sent to the error logs.
P A R T  3

Configuration of Content Services for Standalone Content Engines
CHAPTER 10

Configuring Content Authentication and Authorization on Standalone Content Engines

This chapter describes how to configure content authentication and authorization on standalone Content Engines that are running the ACNS 5.4.1 software and later releases. Content authentication and authorization controls whether an end user (for example, a client browser that issued an HTTP request or an FTP client that issued a nontransparent FTP native request) can access the requested content that is served through the Content Engine.

Note

In the ACNS 5.2.1 software and later releases, HTTP request authentication is supported. Throughout this chapter, the term HTTP request is used to refer collectively to requests over HTTP that include HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests. In the ACNS 5.4.1 software and later releases, proxy authentication of nontransparent FTP native requests is also available.

This chapter contains the following sections:

- About Authentication and Authorization of Content Requests, page 10-2
- Configuring Pass-Through Authentication for WMT Requests, page 10-7
- Configuring End-to-End Authentication for HTTP Requests, page 10-8
- Configuring Request Authentication for HTTP Requests, page 10-10
- Configuring an Authentication Service on Standalone Content Engines, page 10-18
- Configuring the LDAP Acceptable Use Policy Feature, page 10-54
- Configuring Request Authentication for Nontransparent FTP Native Requests, page 10-55

Note

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication. For information about configuring content authentication, authorization, and accounting for Content Engines that are registered with a Content Distribution Manager, see the ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Another type of authentication and authorization method, login authentication and authorization, is used to configure the Content Engine to control the privilege levels that are granted to an administrative user who logs in to the Content Engine for administrative purposes (configuring, monitoring, or troubleshooting the Content Engine). Content authentication and authorization is independent of login authentication and authorization. For information about administrative login authentication and authorization, see Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines.”

### About Authentication and Authorization of Content Requests

As organizations extend the use of web applications and Internet access to their employees, they are confronted with the following challenges:

- How to manage employee use of the Internet
- How to restrict access to online content

Organizations can use content authentication and authorization to address these concerns. Content authentication and authorization can be implemented through various protocols, as summarized in Table 10-1.

#### Table 10-1  
<table>
<thead>
<tr>
<th>Method</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass-through authentication for WMT requests</td>
<td>See the “Configuring Pass-Through Authentication for WMT Requests” section on page 10-7.</td>
</tr>
<tr>
<td>End-to-end authentication for HTTP requests</td>
<td>See the “Configuring End-to-End Authentication for HTTP Requests” section on page 10-8.</td>
</tr>
<tr>
<td>Basic</td>
<td>See the “Basic End-to-End Authentication” section on page 10-8.</td>
</tr>
<tr>
<td>Microsoft NTLM</td>
<td>See the “NTLM End-to-End Authentication” section on page 10-9.</td>
</tr>
<tr>
<td>Request authentication for HTTP requests</td>
<td>See the “Configuring Request Authentication for HTTP Requests” section on page 10-10.</td>
</tr>
<tr>
<td>RADIUS</td>
<td>See the “Configuring the RADIUS Authentication Service” section on page 10-19.</td>
</tr>
<tr>
<td>TACACS+</td>
<td>See the “Configuring the TACACS+ Authentication Service” section on page 10-20.</td>
</tr>
<tr>
<td>LDAP</td>
<td>See the “Configuring the LDAP Authentication Service” section on page 10-21.</td>
</tr>
<tr>
<td>NTLM</td>
<td>See the “Configuring the NTLM Authentication Service” section on page 10-38.</td>
</tr>
<tr>
<td>Group-based authorization for HTTP requests</td>
<td>See the “Configuring Group-Based Authorization for HTTP Requests” section on page 10-46.</td>
</tr>
<tr>
<td>Request authentication for nontransparent FTP native requests</td>
<td>See the “Configuring Request Authentication for Nontransparent FTP Native Requests” section on page 10-55.</td>
</tr>
<tr>
<td>RADIUS</td>
<td>See the “Configuring the RADIUS Authentication Service” section on page 10-19.</td>
</tr>
<tr>
<td>TACACS+</td>
<td>See the “Configuring the TACACS+ Authentication Service” section on page 10-20.</td>
</tr>
<tr>
<td>LDAP</td>
<td>See the “Configuring the LDAP Authentication Service” section on page 10-21.</td>
</tr>
<tr>
<td>NTLM</td>
<td>See the “Configuring the NTLM Authentication Service” section on page 10-38.</td>
</tr>
</tbody>
</table>
In the ACNS 5.2.1 software and later releases, HTTP request authentication is supported. (The term HTTP request is used to refer collectively to requests over HTTP that include HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests.) For more information, see the “Configuring Request Authentication for HTTP Requests” section on page 10-10.

In the ACNS 5.4.1 software and later releases, proxy authentication of nontransparent FTP native requests is also available. This feature provides authentication of nontransparent FTP native requests from such FTP clients as a Reflection X client or a UNIX command line program by the Content Engine that is acting as the FTP proxy. For more information, see the “Configuring Request Authentication for Nontransparent FTP Native Requests” section on page 10-55.

**Note**

In ACNS 5.4.1 software and later releases, you can use IP access control lists (ACLs) to control access to the native FTP proxy service that is running on a standalone Content Engine. For more information, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.

The same process is used to enable and configure RADIUS, TACACS+, NTLM, or LDAP authentication for HTTP requests and nontransparent FTP native requests (for example, you can configure the RADIUS server settings on the Content Engine and enable RADIUS authentication on the Content Engine). However, the following restrictions apply to FTP native caching support:

- No support for FTP request proxy rules
- No support for any URL filtering schemes (good list, bad list, N2H2, Websense, and SmartFilter)

The preceding two restrictions do not apply to HTTP caching support.

### About End-to-End and Pass-through Authentication

End-to-end authentication is authentication that occurs between the client and the origin server. Pass-through authentication is how the Content Engine handles end-to-end authentication. With pass-through authentication, the Content Engine passes through the authentication request and response between the client and the origin server without examining such requests and responses.

### About NTLM Support on Content Engines

Windows NT LAN Manager (NTLM) is the authentication protocol that is used by Microsoft’s browsers (Internet Explorer), proxies, and web servers (IIS). The NTLM protocol is a challenge-response-based protocol and can be used to authenticate and block user access to the Internet. Because NTLM is a challenge-response-based authentication scheme, the clients encrypt the server challenge with their password hash and send the response to the server for validation. The main advantage of using NTLM for HTTP request authentication is that NTLM sends the password in an encrypted format to the server, which originated the authentication challenge. Consequently, the clients can prove their identity without sending a password over the network to the server that originated the authentication challenge.

Typically, enterprises are already using NTLM to enforce access control to information that is stored on their intranet sites. Additionally, enterprises want to protect Internet browsing but not have to prompt their end users for usernames and passwords. NTLM provides this authentication scheme through Microsoft Internet Explorer and domain controllers (DCs). Content Engines support NTLM HTTP request authentication in order to support both of these models. A client (web browser) attempts to perform NTLM HTTP request authentication with the Content Engine in order to be allowed to use the Content Engine (the HTTP proxy server) to access the requested content.
NTLM support on a standalone Content Engine includes the following four types of support: (1) NTLM end-to-end authentication support, (2) NTLM authentication of HTTP requests, and (3) NTLM group information query for authorization purposes, and (4) NTLM support of preloading NTLM-protected objects. See Table 10-2 for a summary of the types of NTLM support on standalone Content Engines that are running the ACNS 5.x software.

**Table 10-2 Summary of NTLM Support on Content Engines**

<table>
<thead>
<tr>
<th>Type of NTLM Support</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTLM authentication of HTTP requests</td>
<td>For HTTP request authentication, the ACNS 5.3.x and earlier releases only support NTLMv1. (NTLMv1 is also referred to as simply “NTLM”.) In the ACNS 5.4.1 software and later releases, both NTLMv1 and NTLMv2 are supported for HTTP request authentication. For more information, see the “Configuring the NTLM Authentication Service” section on page 10-38.</td>
</tr>
<tr>
<td>NTLM group information query for authorization purposes</td>
<td>In the ACNS 5.x software releases, both NTLMv1 and NTLMv2 are supported for NTLM group information query for authorization purposes. For more information, see the “Configuring Group-Based Authorization for HTTP Requests” section on page 10-46.</td>
</tr>
<tr>
<td>NTLM end-to-end authentication</td>
<td>In the ACNS 5.x software releases, both NTLMv1 and NTLMv2 are supported for NTLM end-to-end authentication. For more information, see the “NTLM End-to-End Authentication” section on page 10-9.</td>
</tr>
<tr>
<td>NTLM support of preloading NTLM-protected objects</td>
<td>In the ACNS 5.1.1 software and later releases, NTLMv1 support of preloading NTLM authenticated objects on standalone Content Engines is available. In the ACNS 5.4.1 software release, NTLMv2 support of preloading NTLM authenticated objects was added. For more information, see the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2.</td>
</tr>
</tbody>
</table>

The `ntlm version` global configuration command, which was added in the ACNS 5.4.1 software release, has two command options: the `version 1` option to reenable NTLMv1, and the `version 2` option to enable NTLMv2 on a standalone Content Engine. The `version 1` command option is the default option.

For Content Engines that are registered with a Content Distribution Manager, NTLMv2 support for acquisition and distribution of NTLM-protected content was also added in the ACNS 5.4.1 software release. For information about configuring Content Engines that are registered with a Content Distribution Manager, see the *ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*. 
About NTLM Support

In the ACNS 5.4.1 software and later releases, both NTLMv1 and NTLMv2 are supported for NTLM HTTP request authentication. In the ACNS 5.3.x software and earlier releases, only NTLMv1 is supported for NTLM HTTP request authentication (that is, NTLM request authentication for requests over the HTTP protocol).

With NTLMv1, when a user logs in to a Windows NT or a Windows 2000 domain and starts a browser, the authentication 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 Content Engine, which in turn 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 Content Engine authentication cache. Future requests from this IP address are not challenged until the authentication cache entry expires or is cleared.

NTLMv2 is the successor of NTLMv1. NTLMv2 is designed to be more secure than NTLMv1. The algorithm that is used by NTLMv2 to create the password hash, which is used to encrypt the challenge, is more complex than the algorithm used by NTLMv1. The NTLMv2-encrypted response also includes a timestamp to prevent replay attacks.

In the ACNS 5.4.1 software release, the ACNS software was enhanced to support NTLMv2. In the ACNS 5.3.x software and earlier releases, client authentication will fail if the client attempts to use NTLMv2.

NTLMv2 is not negotiated between the browser and the server (or the proxy). On Windows-based clients and servers, you enable NTLMv2 by modifying a registry variable on the client and the server. When an HTTP response to the client indicates that the server (or the proxy) requires authentication and NTLM is accepted, the client generates a different NTLM response based on the client configuration:

- If NTLMv1 is configured on the client, the client generates an NTLMv1 response when it receives the server challenge.
- If NTLMv2 is configured on the client, the client generates an NTLMv2 response when it receives the server challenge.

Pass-through Authentication Support for NTLM-Enabled Browsers

If the Content Engine is running the ACNS 5.4.1 software or a later release and the client browser is configured for NTLMv2, then pass-through authentication and NTLM HTTP request authentication are both supported.

For pass-through authentication, the following process occurs:

1. After the client browser receives an NTLM challenge from the Content Engine, the client browser generates an NTLMv2 response and sends it to the Content Engine.
2. The Content Engine performs the pass-through authentication, and forwards the NTLMv2 response to the domain controller that sent the authentication challenge to the Content Engine.
3. The Content Engine grants or denies the client access to the requested content based on the domain controller decision.

For an NTLM HTTP request authentication, after the user is validated by the domain controller, the Content Engine saves the information about the client’s username and domain name in its authentication cache so that it will not challenge future requests from the same IP address.
Chapter 10     Configuring Content Authentication and Authorization on Standalone Content Engines

About Authentication and Authorization of Content Requests

Basic Authentication Support for Non-NTLM Enabled Browsers

With non-NTLM enabled browsers (that is, browsers that do not support NTLMv1 or NTLMv2) the Content Engine will only use NTLMv2 to communicate with the domain controller if all of the following conditions exist:

- The Content Engine has been configured to use NTLMv2 (that is, you have specified the `v2` command option for the `ntlm server host` global configuration command).
- The Content Engine is running the ACNS 5.4.1 software or a later release.
- Basic authentication has not been disabled on the Content Engine (that is, you have not disabled basic authentication on the Content Engine by entering the `no ntlm basic-auth enable` global configuration command).

In this situation, when the client browser receives an HTTP response from the Content Engine indicating that client authentication is required, the client browser will perform basic authentication instead of NTLM authentication. The client browser sends the user credentials to the Content Engine in clear text over the network. The Content Engine requests the challenge from the domain controller, generates an NTLMv2 response using the password, and then sends the NTLMv2 response to the domain controller for validation.

Note Even though the client browser sends the Content Engine its password as clear text, the Content Engine does not forward the clear-text password over the network to the domain controller.

About NTLM v2 Support for Request Authentication

By default, the Content Engine uses NTLMv1 instead of NTLMv2 when communicating with each of these eight configured NTLM servers during the request authentication process unless you enable the NTLMv2 feature.

In the ACNS 5.4.1 software release, NTMLv2 support for request authentication of HTTP requests was added. Because NTLMv2 is not negotiated between the client and the server (it is configured separately on both the client and server by changing the Windows registry on the client and the server), there is no configuration parameters that you need to specify on the Content Engine in order for the Content Engine to support NTLMv2. However, if you want the Content Engine to support NTLMv2 when communicating with any of the configured NTLM servers (that is, any of the NTLM servers that have been added to the `host list` through the `ntlm server host` global configuration command), you must enter the `ntlm version 2` global configuration command on the Content Engine:

```
ContentEngine(config)# ntlm version 2
```

The `ntlm version` global configuration command, which was added in the ACNS 5.4.1 software release, has two command options: the `version 1` option to re-enable NTLMv1, and the `version 2` option to enable NTLMv2 on a standalone Content Engine. The `version 1` option is the default option.

It is very important that you specify that the Content Engine use NTLMv2 when communicating with any of the configured NTLM servers, especially for basic authentication, because with basic authentication the Content Engine is actually generating the NTLMv2 response and is communicating directly with the NTLM server.

Note The default behavior for Microsoft Windows clients is to send the LM and NTLM responses, and by default a Microsoft Windows server will accept these responses. If you want to change the default behavior of either the Microsoft Windows client or server (for example, you want the client and the server to send and receive NTLMv2 responses instead of LM and NTLM responses [NTLM version 1 is
also sometimes referred to as simply NTLM), then you must modify the Windows registry.

For instance, you must change the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA\LMCompatibility variable on Windows 9.x systems, or you must change the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA\LMCompatibilityLevel variable on Windows NT and 2000 systems. For information about how to change a Microsoft Windows client registry or a Windows server registry server, refer to your Microsoft Windows documentation.

**Note** If the domain controller has been configured to support only NTLMv2 for security purposes, you must enter the `ntlm version 2` command on the Content Engine to configure the Content Engine to use NTLMv2.

For more information about configuring a list of NTLM servers for a standalone Content Engine, see the “Configuring the NTLM Authentication Service” section on page 10-38.

### Configuring Pass-Through Authentication for WMT Requests

In the ACNS 5.2.1 to ACNS 5.4.x software, pass-through authentication support for WMT MMS requests from WMT clients is available. In the ACNS 5.3.1 software and later releases, pass-through authentication support for WMT RTSP requests from Windows Media 9 players is also available. With this support, the Content Engine establishes a virtual connection between the client and the origin server in order for the origin server to authenticate the client. The Content Engine does a pass through and does not function as the proxy authentication server. Consequently, proxy server authentication is not supported for WMT requests.

**Note** End-to-end authentication is authentication that occurs between the client and the origin server. Pass-through authentication is how the Content Engine handles end-to-end authentication. With pass-through authentication, the Content Engine passes through the authentication request and response between the client and the origin server without examining such requests and responses.

Pass-through authentication for WMT live streams is supported with the following exceptions and requirements:

- Pass-through authentication for a WMT managed live program is not supported. (Managed live programs can be configured in centrally managed deployments only.)
- Pass-through authentication for a static broadcast alias on a Content Engine or for a broadcast point on a Windows Media server is supported. However, when you configure the static broadcast alias on the Content Engine, the client URL to the alias and the source URL to the server must use the same protocol. ACNS software does not support cross-protocol authentication for WMT streams.

In proxy mode, the Windows Media 9 server supports two authentication mechanisms for pass-through authentication of WMT requests:

- Anonymous authentication
- Network authentication
  - Negotiate plug-in (NTLM or Kerberos) authentication
  - Digest plug-in authentication
Configuring End-to-End Authentication for HTTP Requests

End-to-end authentication is authentication that occurs between the client and the origin server. The HTTP protocol has three ways to authenticate a web client that has issued an HTTP request:

- Basic mode
- NTLM mode
- Microsoft Active Directory (AD)/Kerberos

Note

RTP/RTSP supports basic mode for client authentication.

The ACNS 5.2.1 software and later releases support all three modes of client authentication (basic, NTLM, and Active Directory/Kerberos) for pass-through authentication.

End-to-end NTLM authentication using the NTLM method includes pass-through servicing and the caching of web objects that require NTLM authentication.

Basic End-to-End Authentication

The ACNS software can strip NTLM authentication headers to allow fallback to a basic-style authentication challenge against Microsoft IIS servers. Basic authentication is designed to allow browsers to authenticate entered user IDs 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 to the basic authentication method. If the Microsoft IIS server is configured to still accept basic authentication, IIS authentication credentials can proceed through a Content Engine, but with reduced security. Basic authentication is less secure than NTLM authentication because it transmits the user credential information in clear text format.

To configure a standalone Content Engine to strip NTLM headers, enter the `http authenticate-strip-ntlm` global configuration command.

Basic end-to-end authentication support also includes the ability to configure the Content Engine to cache web objects that are authenticated with the basic authentication method. By default, the Content Engine does not cache such objects. However, you can configure the Content Engine to cache objects that are authenticated with the basic authentication method by entering the `http cache-authenticated basic` global configuration command.
NTLM End-to-End Authentication

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

- **NTLM pass-through service**
  The Content Engine sets up a secure persistent connection between the client and the server. 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 Content Engine 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 content request from a client already connected with the intended NTLM server, the Content Engine 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.

**Note**

For end-to-end NTLM authentication, both NTLM Version 1 (NTLMv1) and Version 2 (NTLMv2) are supported.

In the following example, the Content Engine is configured for end-to-end NTLM authentication. By default, basic and NTLM authenticated objects are not cached.

```
ContentEngine(config)# no http authenticate-strip-ntlm
ContentEngine(config)# http cache-authenticated ntlm
ContentEngine# show http cache-authenticated ntlm
NTLM authenticated objects are cached.
```
Configuring Request Authentication for HTTP Requests

Organizations can use HTTP request authentication (content authentication) as a way to restrict access to online content. If HTTP authentication is configured on a standalone Content Engine, the Content Engine checks with an external authentication, authorization, and accounting (AAA) server for user password authentication to determine if the user should be granted or denied access to the requested content. (See Figure 10-1.)

Figure 10-1 HTTP Request Authentication and Group-Based Authorization

ACNS 5.x supports the Lightweight Directory Access Protocol (LDAP), Microsoft NT LAN Manager (NTLM), RADIUS, and TACACS+ protocols, which are used by common AAA servers.

Request authentication and authorization pertain to end user requests that are going through the Content Engine. With request authentication and authorization, the Content Engine is verifying the end user. In contrast, end-to-end authentication and caching of authenticated objects deals with authentication for a particular object, and it is the origin server and not the Content Engine that verifies the end user.
Chapter 10      Configuring Content Authentication and Authorization on Standalone Content Engines

Configuring Request Authentication for HTTP Requests

Controlling Content Access for HTTP Requests

For request authentication, ACNS supports LDAP, NTLM, TACACS+, and RADIUS. For request authorization, the ACNS 5.x software supports group-based access lists. In the ACNS 5.2.1 software release, group-based rules were also added and can be used for group-based authorization. With these features, you can require that end users who are making an HTTP request must be authenticated by an external AAA server, and then authorized by the configured access lists or Rules Template.

The Rules Template feature allows you to configure a set of rules, each clearly identified by an action and a pattern or a group of patterns that a standalone Content Engine uses to filter HTTP requests (HTTP traffic) as well as HTTPS and RTSP traffic. If you have enabled rules processing on a Content Engine (that is, enabled the Rules Template feature on the Content Engine and configured rules for the Content Engine), the Content Engine checks every incoming client request to determine if a rule pattern matches the requested content. If a rule pattern matches the given request, the Content Engine uses the specified action (policy) to handle this incoming traffic.

In the ACNS 5.2 software release, three new rule patterns were added (groupname, username, and groupname-regex). These new rule patterns support access control policies that are based on the group name and username of the authenticated NTLM and LDAP users. Rules based on group names apply to users who have been authenticated through NTLM and LDAP. Rules based on usernames apply to users who are authenticated through LDAP, NTLM, RADIUS, and TACACS+, which are request authentication methods that involve a username for authentication.

For example, the following shows how to enable rule processing on the Content Engine using the rule enable global configuration command and then configure the Content Engine to block all end users in the Engineering group from downloading FTP URLs (FTP requests from a client browser) that contain the expression “java”:

```
ContentEngine(config)# rule enable
ContentEngine(config)# rule pattern-list 1 group-type and
ContentEngine(config)# rule pattern-list 1 groupname Engineering
ContentEngine(config)# rule pattern-list 1 url-regex java
ContentEngine(config)# rule action block pattern-list 1 protocol ftp
```

Another area of access control is the caching of authenticated content. That is, if the website requires client authentication before passing an object to the client, and the object is cached on the Content Engine, then the Content Engine should also authenticate the client before delivering the object to another client. This type of object is called authenticated content.

The following types of content access controls for HTTP requests are supported by standalone Content Engines:

- Authenticating Web Clients Through Pass-Through Authentication, page 10-12
- Authenticating Web Clients Through HTTP Request Authentication, page 10-12
- Configuring Group-Based Access Lists on Standalone Content Engines, page 10-46
Authenticating Web Clients Through Pass-Through Authentication

The HTTP protocol itself has three ways to authenticate end users (web clients) who issue an HTTP request for content that is served through the Content Engine:

- Basic mode
- NTLM mode
- Microsoft Active Directory (AD)/Kerberos

**Note**
RTP/RTSP supports basic mode for client authentication.

Pass-through authentication is used when clients request access to content on a website that requires clients to authenticate themselves (that is, clients must enter their username and password in a popup login window) before the content is sent to requesters. The Content Engine, which is between the client and the website (either through proxy-configuration or transparent interception methods), should not hinder this client authentication. In order to support this authentication exchange between the client and the web server, the Content Engine must pass the authentication exchange between the client and the web server (it must tunnel the authentication exchanges).

Occasionally a website uses the client’s IP address to authenticate a client. This is typically used in older web servers and is not the preferred solution for client authentication. However, for such older websites there must be a way for the Content Engine “to get out of the way” between the client and the web server so that the client can be authenticated. The authentication traffic bypass feature can be used in these cases. For information on this topic, see the “Configuring Authentication Traffic Bypass on Standalone Content Engines” section on page 15-4.

**Note**
The Content Engine must also guarantee that if it caches any objects that required client authentication (that is, it caches authenticated content), then it will not deliver cached authenticated content to other clients unless those clients are authenticated to access the cached content.

Authenticating Web Clients Through HTTP Request Authentication

HTTP request authentication is used when the Content Engine communicates with an external AAA server to authenticate the client who is requesting content. This type of authentication is used to allow or prevent a client from accessing the Internet. For example, the BigCorp company may want to limit Internet access to its employees and not allow their temporary employees to access the web.

In this case, the Content Engine is located at the Internet gateway of the BigCorp company and can be used to enforce this policy. When the Content Engine receives a client request to access content that is served through the Content Engine, the following occurs:

1. The Content Engine sends an authentication challenge to the client, and prompts the client to enter authentication information such as the username and password.
2. The Content Engine communicates with the AAA server to determine if the supplied authentication information is valid.
3. Based on the responses from the AAA server, the following occurs:
   a. If the AAA server validates the user, the Content Engine allows the request to go through (that is, it allows the client to access the requested content).
   b. If the AAA server does not validate the user, the Content Engine denies the request and sends the client an authentication failed message.

In the ACNS 5.x software, you can specify which groups of users are allowed to access the Internet and which groups are not allowed. In this case, the Content Engine asks the AAA server not only whether clients are who they claim to be but also which groups they belong to. The Content Engine then performs the appropriate actions based on the responses from the AAA server. This type of access control is referred to as **HTTP request authentication**, and controls what content the client can access through the Content Engine. The ACNS 5.x software supports the LDAP, NTLM, RADIUS, and TACACS+ protocols, which are used by common AAA servers. (See Table 10-1.)

In the case of NTLM, HTTP request authentication authenticates a user’s domain, username, and password with a preconfigured primary domain controller (PDC) before allowing the requests from the user to be served by the Content Engine.

In the ACNS 5.2 software release, the ability to specify the list of domains that are allowed to perform NTLM HTTP request authentication through the Content Engine was added.

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 the authentication server database. When the Content Engine authenticates a user through an authentication 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 authentication server lookups.

The Content Engine supports HTTP request authentication for both proxy mode and transparent (WCCP) mode access.
   - In proxy mode, the Content Engine uses the client’s user ID (UID) as a key for the Content Engine’s authentication cache for TACACS+, LDAP, and RADIUS authentication methods.
   - In transparent mode for all methods and in proxy mode for NTLM, the Content Engine uses the client’s IP address as a key for the Content Engine’s authentication cache.

If you are using HTTP request authentication in transparent mode or NTLM in either transparent or proxy mode, we recommend that the AuthTimeout interval configured with the `http authentication cache timeout` global configuration command be short, because the key for the Content Engine’s authentication cache is the client’s IP address. IP addresses can be reallocated, or different users can access the Internet through an already authenticated device (such as a PC workstation). Shorter AuthTimeout values help reduce the possibility that individuals can gain access using previously authenticated devices. In the ACNS 5.2 software release, an absolute timeout configuration option was introduced with the `http authentication cache ttl` global configuration command. This absolute timeout can also be configured to help reduce the possibility of individuals gaining access by using previously authenticated browsers. For more information, see the “Specifying a Reauthentication Interval” section on page 7-14.
Understanding Proxy Server Mode HTTP Request Authentication

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

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 eight 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 authentication success, CE2 either serves the request out of its cache or forwards the request to the origin server. (This credential forwarding capability is not configured by default. If you want credential forwarding, you must explicitly configure it through the `http append proxy-auth-header host CE2ipaddress` global configuration command.)

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.

If you want to avoid authentication on an upstream Content Engine after authentication is performed downstream, you can use the `rule no-auth` global configuration command to exclude the downstream Content Engine IP address.

When the Content Engine is operating in proxy server mode and is configured for HTTP request authentication, the following events occur if one of the following two situations is true: (1) the Content Engine receives a proxy-style request directly from a client, or (2) the Content Engine receives a WCCP-redirected request and the Content Engine `http authentication header` global configuration command option is set to 407 (Proxy Authorization Required) 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 message (Proxy Authorization Required) 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 message to the client.

**Understanding Transparent Mode HTTP Request Authentication**

When the Content Engine is operating in transparent mode, the user IP address is used as a key to the authentication cache. The Content Engine will always first look for an X-Forwarded-For header, then the source IP address.

If there are two levels of Content Engines in a proxy chain (CE1 at the first level [the Content Engine that is nearest to the client] and CE2 at the second level), and CE1 and CE2 both have the `http append x-forwarded-for-header multiple-ip-address` global configuration command configured on them, then the following will occur:

1. After receiving a request from the client, CE1 will append the default client’s IP address to the X-Forwarded-for header and then forward the request to CE2. For example, if the client’s IP address is 10.1.1.20, the X-Forwarded-For header would be “X-Forwarded-for: 10.1.1.20”).

2. After CE2 receives the request from CE1, CE2 will append the IP address of CE1 to the X-Forwarded-for header. Consequently, the X-Forwarded-For header will contain the client’s IP address (which is already present in the header) and the CE1’s IP address separated by comma. For example, if the IP address of CE1 is 10.40.1.40, the X-Forwarded-For header would be “X-Forwarded-for: 10.1.1.20, 10.40.1.40”.

In the ACNS 5.4.1 software and later releases, multiple IP addresses in the X-Forwarded-For header are supported. Enter the `http append x-forwarded-for-header multiple-ip-address` global configuration command to enable support for appending multiple IP addresses to the X-Forwarded-for header. If you specify this command and if the request arrives at CE2 from CE1 with an X-Forwarded-For header, then CE1’s IP address is appended to the existing client’s IP address in the X-Forwarded-For header separated by a comma.

**Note**

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.

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 (Proxy Authorization Required) 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.
When the Content Engine is operating in transparent mode and is configured for HTTP request authentication, the following events occur if either of the following is true: (1) the Content Engine receives a redirected request from a client, or (2) the \texttt{http authentication header} global configuration command option is set to 401 (Unauthorized) 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 Content Engine authentication cache.

In the ACNS 5.4.1 software and later releases, multiple IP addresses in the X-Forwarded-For header are supported.

You can configure one of the following authentication and authorization services to control HTTP request authentication on a standalone Content Engine:

- Configuring the RADIUS Authentication Service, page 10-19
- Configuring the TACACS+ Authentication Service, page 10-20
- Configuring the LDAP Authentication Service, page 10-21
- Configuring the NTLM Authentication Service, page 10-38

\textbf{Note} NTLM support on the Content Engine includes the following three types of support: (NTLM end-to-end authentication support, NTLM authentication of HTTP requests, and NTLM group information query for authorization purposes. See Table 10-2 for a summary of the types of NTLM support on standalone Content Engines that are running the ACNS 5.x software.
Guidelines for Configuring Request Authentication for HTTP Requests

When configuring HTTP request authentication on a standalone Content Engine, remember the following important points:

- Only one request authentication scheme can be enabled on the Content Engine at a time.
- 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. For information about adjusting the size of the authentication cache, see the “Configuring Authenticated HTTP Cache Settings” section on page 7-12.
- As long as the authentication entry is retained, subsequent attempts to access restricted Internet content by that user do not require server lookups. The http authentication cache timeout global configuration 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 client reauthentication.
- For security purposes, the ability to specify an absolute Time To Live (TTL) for HTTP request authentication cache entries was added in the ACNS 5.2 software release. For more information, see the “Specifying a Reauthentication Interval” section on page 7-14.
- To exclude domains from HTTP request authentication, use the rule no-auth domain global configuration command. TACACS+, NTLM, RADIUS, or LDAP authentication takes place only if the site requested does not match the specified pattern.
- For additional security, when using NTLM for HTTP request authentication, use the no ntlm basic-auth enable global configuration command. This command prevents the Content Engine from offering the basic authentication method to the client, or prevents the Content Engine from honoring a basic authentication request from the client.

Note HTTP authentication featuring RADIUS and LDAP existed in Cache software 2.x releases and was configured through the radius-server and ldap commands, respectively.

For ACNS 5.x 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 option 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.

The Content Engine uses simple (nonencrypted) authentication to communicate with an LDAP authentication server.
Authentication servers can be specified with the `host` option for NTLM, RADIUS, and LDAP servers, or `server hostname` option for TACACS+ servers.

- NTLM allows up to eight authentication servers for HTTP request authentication. The order of server configuration determines the order of load balancing or failover. For example, the first server configured (server 1) is the primary server and is sent all of the requests first. The last server configured (server 8) is the last server that the Content Engine contacts.
- LDAP allows two authentication servers to be specified.
- RADIUS allows five authentication servers to be specified.
- TACACS+ allows three authentication servers to be specified.

These additional authentication servers act as backup authentication servers and will only be used when the primary authentication server is not available. If the Content Engine cannot connect to any of the authentication servers, no authentication takes place, and users who have not been previously authenticated are denied access.

Once a user has been authenticated through a TACACS+, LDAP, NTLM, or RADIUS server, all transaction logs generated by the Content Engine for that user contain user information if a transaction log format containing the username is configured (Extended-Squid format or custom format with `%u` in the format string).

- 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` global configuration command is specified, the user information is suppressed. For more information on transaction logging, see Chapter 21, “Monitoring Standalone Content Engines and Transactions.”

The following are some examples of using the Content Engine CLI to configure HTTP request authentication on a standalone Content Engine:

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

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

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

```plaintext
ContentEngine(config)# radius-server 172.16.90.121
```

### Configuring an Authentication Service on Standalone Content Engines

For information about how to configure an authentication service on a standalone Content Engine, see the following section:

- Configuring the RADIUS Authentication Service, page 10-19
- Configuring the TACACS+ Authentication Service, page 10-20
- Configuring the LDAP Authentication Service, page 10-21
- Configuring the NTLM Authentication Service, page 10-38
Configuring the RADIUS Authentication Service

RADIUS authentication clients reside on the Content Engine running ACNS 5.x software. When enabled, these clients send authentication requests to a central RADIUS server that contains user authentication and network service access information.

When the Content Engine communicates with a RADIUS authentication server, the RADIUS server requires that the RADIUS client (Content Engine) to be registered. Additionally, the RADIUS client and server use the secret key. This secret key has to be configured on both the RADIUS server and RADIUS client (Content Engine); the RADIUS client uses this RADIUS key to encrypt and decrypt authentication packets.

In the ACNS 5.4.1 software and later releases, RADIUS authentication for nontransparent FTP native requests is supported. The same process is used to enable and configure RADIUS authentication for HTTP requests and nontransparent FTP native requests. However, the following restrictions apply to FTP native caching support:

- No support for FTP request proxy rules
- No support for any URL filtering schemes (good list, bad list, N2H2, Websense, and SmartFilter)

Note

No RADIUS authentication will be performed if no RADIUS servers are configured, as described in the “Specifying RADIUS Authentication Settings for Standalone Content Engines” section on page 17-10.

The following is an example of how to configure RADIUS authentication for HTTP requests on a standalone Content Engine:

Step 1
Configure the RADIUS server settings on the Content Engine, as described in the “Specifying RADIUS Authentication Settings for Standalone Content Engines” section on page 17-10.

Step 2
Enable RADIUS authentication for HTTP requests on the Content Engine:

- From the Content Engine GUI, choose Caching > RADIUS to display the RADIUS window. Click the Enable RADIUS On radio button to enable RADIUS authentication on this Content Engine. Click Update to save the settings.
- From the Content Engine CLI, use the radius-server global configuration command.

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

The following example enables the RADIUS client on the Content Engine, specifies an external RADIUS server, specifies the RADIUS key, accepts retransmit defaults, and excludes the domain name mydomain.net from RADIUS authentication:

```
ContentEngine(config)# radius-server enable
ContentEngine(config)# radius-server host 172.16.90.121
ContentEngine(config)# radius-server key myradiuskey
ContentEngine(config)# rule enable
ContentEngine(config)# rule no-auth domain mydomain.net
```
**Chapter 10   Configuring Content Authentication and Authorization on Standalone Content Engines**

**Configuring an Authentication Service on Standalone Content Engines**

**Note**  The **rule** command is relevant to RADIUS only if the **radius-server redirect** option has been configured.

The configuration is displayed and verified with the **show radius-server** and **show rule all** EXEC commands:

```
ContentEngine# show radius-server
Radius Configuration:
---------------------
Radius Authentication is on
  Timeout     = 5
  Retransmit  = 3
  Key         = ****
  Servers
  ------
  IP 172.16.90.121 Port = 1645  State: ENABLED

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

**Note**  To disable RADIUS authentication on the Content Engine, use the **no radius-server enable** global configuration command.

**Configuring the TACACS+ Authentication Service**

When the Content Engine communicates with a TACACS+ authentication server, the same secret key (for example, tackey) has to be configured on the TACACS+ server and each of the TACACS+ clients (Content Engine, routers, and so forth) that want to communicate with this TACACS+ server.

```
ContentEngine(config)# tacacs key "tackey"
```

The TACACS+ server uses this common client key to encrypt or decrypt. The TACACS+ authentication server does not require that the TACACS+ client be registered.

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. The ACNS 4.1 software and later releases support TACACS+ only and not TACACS or Extended TACACS.

In the ACNS 5.4.1 software and later releases, RADIUS authentication for nontransparent FTP native requests is supported. The same process is used to enable and configure RADIUS authentication for HTTP requests and nontransparent FTP native requests. However, the following restrictions apply to FTP native caching support:

- No support for FTP request proxy rules
- No support for any URL filtering schemes (good list, bad list, N2H2, Websense, and SmartFilter)
Chapter 10 Configuring Content Authentication and Authorization on Standalone Content Engines

Configuring an Authentication Service on Standalone Content Engines

The following example shows how to configure TACACS+ authentication for HTTP requests on a standalone Content Engine:

**Step 1** Configure the Content Engine to use one or more TACACS+ servers.

ContentEngine(config)# tacacs server ip_addr [primary]

This example shows how to specify a specific TACACS+ server as a primary server:

ContentEngine(config)# tacacs server 172.16.50.1 primary

This example shows how to specify a specific TACACS+ server as a backup server. This can be achieved by not specifying the primary option:

ContentEngine(config)# tacacs server 172.16.50.2

**Step 2** Specify the TACACS+ key.

ContentEngine(config)# tacacs key key

**Step 3** Specify the TACACS+ timeout interval. For example, configure the Content Engine to wait 15 seconds before declaring a timeout if it has not received a response from the TACACS+ server:

ContentEngine(config)# tacacs timeout 15

**Step 4** Specify the TACACS+ retransmit count.

For example, configure the Content Engine to retransmit only one time to the TACACS+ server if a TACACS+ timeout occurs:

ContentEngine(config)# tacacs retransmit 1

**Step 5** Specify the mechanism for TACACS+ password authentication.

For example, use ASCII clear text as the mechanism by entering the ascii keyword:

ContentEngine(config)# tacacs password ascii

**Step 6** Enable TACACS+ authentication on the Content Engine.

ContentEngine(config)# tacacs enable

---

**Note** For more information about a TACACS+ authentication setting (for example, specifying a TACACS+ key), see Table 17-3. For more detailed information about the tacacs server global configuration command, see the Cisco ACNS Software Command Reference Guide, Release 5.5 publication.

### Configuring the LDAP Authentication Service

To address the issue that the X.500 protocol Directory Access Protocol (DAP) was too complex for many directory implementations, the University of Michigan developed the Lightweight Directory Access Protocol (LDAP). LDAP is a directory service protocol that is simpler than the TCP/IP-based version of DAP, and can be used to access information directories.

A standalone Content Engine can be configured to restrict user Internet access by using an LDAP server for authentication purposes, which provides most of the services of the X.500 protocol with less complexity and overhead.
To exclude domains from LDAP authentication, use the `rule no-auth domain` global configuration command. Authentication challenges from LDAP, RADIUS, TACACS+, or SSH take place only if the request does not match the specified `no-auth` pattern.

Typically, an LDAP client (the Content Engine) queries an LDAP server and obtains the user’s credentials such as user’s account expiration, privileges, and group membership from the remote LDAP directory on an OpenLDAP or third-party LDAP server. In the ACNS 5.1 software and later releases, the Content Engine can also authenticate and authorize a user who is configured in a remote Active Directory user database on a Microsoft Active Directory server.

Figure 10-2 shows how the Content Engine (an LDAP client) works with any of the following types of servers to perform LDAP authentication of HTTP requests:

- OpenLDAP servers (shareware servers)
- Third-party LDAP servers (for example, a Sun Microsystems iPlanet server)
- Microsoft Active Directory (AD) servers

**Figure 10-2** shows how the Content Engine performs HTTP request authentication with LDAP if the Content Engine is operating in proxy mode. If the Content Engine were operating in transparent mode, there would also be a WCCP-enabled router between the web clients and the Content Engine (LDAP client).
The ACNS 5.x software supports LDAP Version 2 and Version 3 and supports all LDAP features except for Secure Authentication and Security Layer (SASL). 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.

The Active Directory group attribute is an LDAP Version 3 extension. Consequently, the Content Engine must use LDAP Version 3 to query a Microsoft Active Directory server separately for this information.

Table 10-3 lists the features that are supported on the different types of LDAP servers (shown in Figure 10-2). An “x” indicates support of a particular feature.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Third-Party LDAP Servers</th>
<th>Open LDAP Servers</th>
<th>Microsoft Active Directory Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP Version 2</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>LDAP Version 3</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Organizational units (ou)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Active Directories (AD)</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Static groups</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

In the ACNS 5.4.1 software and later releases, LDAP authentication for nontransparent FTP native requests is supported. The same process is used to enable and configure LDAP authentication for HTTP requests and nontransparent FTP native requests. However, the following restrictions apply to FTP native caching support:

- No support for FTP request proxy rules
- No support for any URL filtering schemes (good list, bad list, N2H2, Websense, and SmartFilter)

When you configure LDAP authentication of HTTP requests and nontransparent FTP native requests, remember these important points:

- You can configure LDAP authentication of HTTP requests and nontransparent FTP native requests through the Content Engine GUI or the CLI.
- From the Content Engine GUI, choose Caching > LDAP. In the displayed LDAP window, click the Enable LDAP On radio button to enable LDAP authentication, and use the window to configure LDAP authentication. For more information on this topic, click the HELP button in the window.
- From the Content Engine CLI, use the ldap server global configuration command.
  - To enable LDAP authentication, use the ldap server enable command
  - To specify the LDAP protocol version to be used, use the ldap server version ver_num global configuration command. The options are Version 2 or Version 3. The following example show how to specify LDAP Version 3:
    ```bash
    ContentEngine(config)# ldap server enable ldap server version 3
    ```
  - To specify the number of seconds that the Content Engine is to wait for a response before timing out on a connection to a particular LDAP server, Use the ldap server timeout seconds global configuration command. The default value is 5 seconds.
To specify the number of connection attempts that are allowed to an LDAP server, use the `ldap server retransmit retries` global configuration command. The default is two attempts. This example shows how to set the LDAP retransmit count to 3:

```
ContentEngine(config)# ldap server retransmit 3
```

To specify the user ID attribute, use the `ldap server userid-attribute useriword` command. By default, `uid` is specified.

To specify the filter string (for example, objectclass=user) to be used in the database search, use the `ldap server filter filterword` global configuration command. There is no default.

To specify the base distinguished name string for the database search, use the `ldap server base baseword` global configuration command. There is no default value for this field. This example shows how to specify the base distinguished name that specifies the starting point for this database search.

```
ContentEngine(config)# ldap server base dc=cisco,dc=com
```

To specify the administrative distinguished name for the database search, use the `ldap server administrative-dn name` global configuration command. There is no default value for this field.

To specify the administrative distinguished password for the database search, use the `ldap server administrative-passwd passwd` global configuration command. There is no default value for this field.

To specify the port number on which the LDAP server is listening, use the `ldap server port port-num` global configuration command. The default port number is 389.

Specify the LDAP server that the Content Engine should use for authentication. (A primary and secondary LDAP server can be specified.)

To designate an LDAP server as the primary server, use the `ldap server host {hostname | hostipaddress} primary` global configuration command. Specify the IP address or hostname of the primary LDAP server.

To designate an LDAP server as the secondary server, use the `ldap server host {hostname | hostipaddress} secondary` global configuration command. Specify the IP address or hostname of the primary LDAP server.

This example shows how to specify two LDAP servers as primary and secondary HTTP request authentication servers:

```
ContentEngine(configure)# ldap server 172.16.1.1 primary
ContentEngine(configure)# ldap server 172.16.1.2 secondary
```

**Note** No LDAP authentication will be performed if no LDAP servers are configured.

To retrieve group names from the LDAP database for group-based authorization, use the `ldap server group` global configuration command. Use this command to instruct the LDAP server, which is performing the database query, to retrieve the names of the groups that the authenticated user belongs to (for example, Marketing or Engineering). The Content Engine uses these retrieved group names to perform LDAP group-based authorization by checking its access lists to determine whether the groups should be granted or denied access to the requested content.
The ACNS 5.1 software and later releases support LDAP Version 2 and Version 3 and supports all LDAP features except for Secure Authentication and Security Layer (SASL).

Table 10-4 lists the Content Engine’s default configuration for LDAP authentication of HTTP requests.

Table 10-4  Default Configuration for LDAP Authentication of HTTP Requests

<table>
<thead>
<tr>
<th>Feature or Setting</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP authentication</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow mode</td>
<td>Disabled</td>
</tr>
<tr>
<td>Base distinguished name</td>
<td>None specified (an empty string)</td>
</tr>
<tr>
<td>Filter for database searches</td>
<td>None specified</td>
</tr>
<tr>
<td>LDAP retransmit attempts</td>
<td>2 times</td>
</tr>
<tr>
<td>LDAP server timeout</td>
<td>5 seconds</td>
</tr>
<tr>
<td>User ID attribute</td>
<td>uid</td>
</tr>
<tr>
<td>Group attribute</td>
<td></td>
</tr>
<tr>
<td>• Organization unit (ou)</td>
<td>Enabled</td>
</tr>
<tr>
<td>• Custom attribute</td>
<td>Disabled</td>
</tr>
<tr>
<td>• Active Directory (memberOf)</td>
<td>Disabled</td>
</tr>
<tr>
<td>Static group database queries</td>
<td>Disabled</td>
</tr>
<tr>
<td>• Group attribute</td>
<td>None specified</td>
</tr>
<tr>
<td>• Group member</td>
<td>None specified</td>
</tr>
<tr>
<td>• Nested groups</td>
<td>Disabled</td>
</tr>
<tr>
<td>• Nested level</td>
<td>1 level</td>
</tr>
<tr>
<td>Administrative distinguished name</td>
<td>None specified</td>
</tr>
<tr>
<td>Administrative password</td>
<td>None specified</td>
</tr>
<tr>
<td>LDAP version</td>
<td>LDAP Version 2</td>
</tr>
<tr>
<td>LDAP port</td>
<td>Port 389</td>
</tr>
<tr>
<td>Policy redirect feature</td>
<td>Disabled</td>
</tr>
<tr>
<td>Password expiry feature</td>
<td>Disabled</td>
</tr>
<tr>
<td>Primary LDAP server</td>
<td>None specified</td>
</tr>
<tr>
<td>Secondary LDAP server</td>
<td>None specified</td>
</tr>
</tbody>
</table>
Example of Configuring LDAP Authentication Service

The following example shows how to use the Content Engine CLI to configure LDAP authentication for HTTP requests and nontransparent FTP native requests on a standalone Content Engine:

**Step 1** Specify the IP address or hostname of the LDAP authentication server that the Content Engine should use to authenticate HTTP requests and nontransparent FTP native requests.

```
ContentEngine(config)# ldap server host 10.1.1.1
```

**Step 2** Configure the Content Engine to use LDAP Version 3 versus LDAP Version 2 (the default), if necessary.

```
Figure 10-2 shows that the LDAP client on the Content Engine can send LDAP Version 2 or Version 3 requests to an OpenLDAP server or a third-party LDAP server. However, the Content Engine must use LDAP Version 3 to communicate with a Microsoft Active Directory server. By default, the Content Engine uses LDAP Version 2. To change this default, use the `ldap server version 3` global configuration command.
```

```
ContentEngine(config)# ldap server version 3
```

**Step 3** By default, LDAP authentication of HTTP requests and nontransparent FTP native requests is disabled on a Content Engine. Enable LDAP authentication on the Content Engine, as follows:

```
ContentEngine(config)# ldap server enable
```

**Step 4** Specify the search criteria.

These search criteria are passed to the LDAP server, which uses these criteria to search through its user database. The LDAP server retrieves the user’s password for authentication. If the user is authenticated, then the LDAP server searches through its database for the requested user membership information, and returns this information to the Content Engine.

Search criteria can include such information as the group attribute (for example, organizationUnit or Active Directory), the name of the user identification (UID) attribute, and the starting point of the search. You can also specify a filter (for example, objectclass=users) to restrict the scope of the database search. If the Content Engine is configured for group-based authorization, then the Content Engine will use the retrieved group names to perform group-based authorization for authenticated users. For more information on group-based authorization through access lists, see the “Configuring Group-Based Authorization for HTTP Requests” section on page 10-46.

To specify the search criteria, you must understand the structure of the user database being queried. For example, the Content Engine by default will request that the authentication server search its database for organizational unit (ou) group membership.

---

**Note**

You can search an OpenLDAP server, a third-party LDAP server, or a Microsoft Active Directory server for organizational unit group membership information. You can only search for Active Directory group membership if the authentication server is a Microsoft Active Directory server.

The organizational unit option and the Active Directory option are independent parameters. You can configure a Content Engine to search a Microsoft Active Directory server database for organizational unit group membership as well as Active Directory group membership.
You can change this default. The following example shows how to configure the Content Engine to query a Microsoft Active Directory server for only the Active Directory group membership and not the organizational unit group membership for authenticated users:

```
Content Engine(config)# ldap server group active-directory enable
```

The following sample output shows that the Content Engine is now configured to search the Microsoft Active Directory server database for Active Directory group membership (memberOf) for authenticated users:

```
ContentEngine# show ldap
LDAP Configuration:
-------------------
LDAP Authentication is enabled
Allow mode:     disabled
Base DN: "DC=cisco,DC=com"
Filter:         <none>
Retransmits:    2
Timeout:        5 seconds
UID Attribute:  sAMAccountname
Group Attribute:
   organizationUnit: disabled (ou)
   Custom Attribute: disabled
   Active Directory: enabled (memberOf)
```

---

**Note** For more information on the structure of the LDAP database, see the “Understanding the Structure of the LDAP Database” section on page 10-28.

---

You can also configure the Content Engine to perform group-based authorization for HTTP requests after a user has been successfully authenticated. If the Content Engine is configured for group-based authorization, then the Content Engine checks its access lists to determine whether the groups that the authenticated user belongs to should be granted or denied access to the requested content. Based on the results of this LDAP group-based authorization, the Content Engine grants or denies user access to the requested content. (See Figure 10-1.)

In order for LDAP group-based authorization to occur, you must have completed the following prerequisite tasks on the Content Engine:

- Enabled and configured group name-based access lists, as described in the “Example of Configuring LDAP Group Authorization with Group-Based Access Lists” section on page 10-47.
- Configured the Content Engine to request that the LDAP server retrieve the names of the groups that the authenticated user belongs to (for example, Marketing or Engineering). You must understand the structure of the user database being queried in order to configure the Content Engine to request this information retrieval. For more information on this topic, see the next section, “Understanding the Structure of the LDAP Database.”
Understanding the Structure of the LDAP Database

In order to configure the LDAP client on the Content Engine to query a remote user database, you must understand the structure of the user database being queried.

LDAP database entries are arranged in a hierarchical directory tree that reflects logical or geographic boundaries. The LDAP directory has the following structure. (See Figure 10-3.)

- The topmost node of the LDAP directory tree is named root.
- Below the root, there are organization nodes (o) for companies, states, or national organizations (for example, o=cisco, o=texas, and o=redcross).
- Below organization nodes, there are LDAP group nodes for such organizational units as departments (for example, ou=Marketing and ou=Engineering) and branch offices.
- At the bottom of the tree, there are individual nodes (cn) for common names of people (for example, cn=Jane Smith), documents, and such shared resources as printers.

**Figure 10-3 LDAP Database Structure**

Because the groups named Hardware Engineers and Software Developers are nested under the parent group named Engineering, they are called nested groups. Nested groups allow the LDAP server administrator to create hierarchical relationships that can be used to define inherited group membership.

An LDAP directory can contain such information as text, photographs (JPEGs), URLs, binary data, and public key certificates.
**About LDAP User Database Entries**

An entry in an LDAP user database (an LDAP directory) is a collection of attributes that has a name, called a distinguished name (dn). Each of the entry’s attributes has a type, name, and one or more values.

- **Attribute type**—An integer, string, or character
- **Attribute name**—Name of the attribute (for example, cn for a common name, givenName for a given name [first name], or mail for an e-mail address)
- **Attribute value**—Value of the attribute (for example, Jane Smith, Jane, or jsmith50@cisco.com)

The following example shows the complete database entry for Jane Smith in an OpenLDAP or third-party LDAP server database:

```
# Jane Smith, cisco, com
dn: cn=Jane Smith,dc=cisco,dc=com
telephoneNumber: (408) 123-9100
mail: jsmith50@cisco.com
uid: jsmith50
givenName: Jane
sn: Smith
cn: Jane Smith
```

The following example shows the complete entry for Jane Smith in a Microsoft Active Directory server database:

```
# Jane Smith, cisco, com
dn: CN=Jane Smith,DC=cisco,DC=com
memberOf: CN=Users,DC=cisco,DC=com
telephoneNumber: (408) 123-9100
mail: jsmith50@cisco.com
uid: jsmith50
givenName: Jane
sn: Smith
cn: Jane Smith
```

An entry in an LDAP user database is referenced by its distinguished name (dn). A distinguished name is constructed by taking the name of the entry itself and concatenating the names of its ancestor entries in the hierarchical directory structure of the LDAP database. For example, if the common name (cn) is Jane Smith and this individual belongs to the organization named cisco, then the distinguished name for this LDAP directory entry is as follows:

- `cn=Jane Smith,dc=cisco,dc=com` for an OpenLDAP server or a third-party LDAP server
- `CN=Jane Smith,DC=cisco,DC=com` for a Microsoft Active Directory server

**About User Groups in an LDAP Directory**

An LDAP directory can contain groups of users that are part of an LDAP group (for example, groups named Marketing and Engineering). An LDAP group is a list, which is a collection of names. An LDAP group has an objectclass attribute of `groupOfNames`, and consists of one or more members. (A group cannot be empty.)

As part of an LDAP database search, the LDAP server can be instructed to retrieve the group names of the authenticated user. The Content Engine uses the retrieved group names and its group name-based access lists to perform group-based authorization for these LDAP users.
Note

LDAP allows you to control which attributes are required and allowed in an entry through the special attribute named `objectclass`. The values of the `objectclass` attribute determine the schema rules the entry must obey. For more details on LDAP, see the RFC 1777 Lightweight Directory Access Protocol.

There are multiple ways to support LDAP grouping in a user database:

- Grouping LDAP Users into Organizational Units, page 10-30
- Grouping LDAP Users into Static Groups, page 10-31
- Grouping Users into Active Directory Groups, page 10-33

The method that the database administrator used to group users in a database determines how you must configure the Content Engine to query that database for user membership information. For examples of how to use the CLI to configure the Content Engine to perform LDAP directory queries, see the “Querying LDAP Servers for User Membership Information” section on page 10-36.

In LDAP Version 3, groups can be defined as static, dynamic, or organizationUnit (organizational units). The groups can be nested. The ACNS 5.1 software and later releases support static and organizationUnit groups but do not support dynamic groups.

**Grouping LDAP Users into Organizational Units**

When the LDAP server administrator groups LDAP users based on organizational units (for example, Marketing), the administrator can assign a user to any group, and can nest groups. An organizational unit is synonymous with a native LDAP group. This method is also referred to as *native LDAP group configuration*. All conventional LDAP Version 3 servers support native LDAP group configuration in the user database.

By default, the Content Engine is configured to query an LDAP server for organizational unit group membership. The following example shows the two entries in the LDAP database that are used to assign the LDAP user named Penny Gold to the organizational unit named Marketing.

In the first database entry, the LDAP server administrator defines the group node for the organizational unit named Marketing.

```
dn: cn=Marketing,dc=cisco,dc=com
cn: Marketing
objectclass: groupOfNames
```

In the second database entry, the administrator defines the individual node for Penny Gold. This node contains all of Penny Gold’s user membership information.

```
# Penny Gold, marketing, cisco, com
dn: cn=Penny Gold,ou=Marketing,dc=cisco,dc=com
telephoneNumber: (408) 123-4444
mail: pgold@cisco.com
uid: pgold
givenName: Penny
sn: Gold
cn: Penny Gold
```

Because Penny Gold’s organizational unit (ou) is specified as Marketing, she is assigned the group privileges of this particular group. If you have configured the Content Engine for group-based authorization (for example, configured an access list that permits the members of the Marketing group to access content that is served through the Content Engine). And when the Content Engine has authenticated Penny Gold, it will grant her access to the requested content because she is a member of the Marketing group.
Based on the above database structure, the following example shows how you would configure the Content Engine to query this database for user membership information for such users as Penny Gold. The server with an IP address of 172.16.1.1 (an OpenLDAP server or the third-party LDAP server) is instructed to start the database search at the node named Marketing, to search for the user identification (username and user password), and to retrieve the group names of the authenticated user.

```
ContentEngine(config)# ldap server 172.16.1.1
ContentEngine(config)# ldap server userid-attribute uid
ContentEngine(config)# ldap server organizationUnit enable
ContentEngine(config)# ldap server base "dc=cisco,dc=com"
ContentEngine(config)# ldap server enable
```

Because the organizationUnit option is enabled, the LDAP server queries the database for the organizationUnit configuration (ou attribute) of the user account. If the user belongs to more than one organizational unit, the LDAP server will return a string that contains all of the group names to which the user belongs. If the Content Engine is configured for LDAP group-based authorization, it uses the retrieved group names and its access lists to perform group-based authorization on the authenticated user.

### Grouping LDAP Users into Static Groups

When the LDAP server administrator uses static groups to group LDAP users in a user database, the administrator explicitly specifies each member of the static group individually. LDAP administrators assign users to an LDAP static group in order to set up a user’s authorization privileges. A user’s access to the Internet is allowed or denied based on the privileges that have been assigned to the static group (for example, Engineering or Hardware Engineers).

A static group defines each member individually using the object class attribute of groupOfNames or groupOfUniqueNames. These object classes require the attribute member (groupOfNames) or uniqueMember (groupOfUniqueNames). There is a one-to-one correspondence between the object class name and the member name attribute. A static group that uses these structural object classes must have at least one member; it cannot be empty.

The Content Engine LDAP static group feature allows you to query both object classes (groupOfNames and groupOfUniqueNames) for group members.

- The following example shows how to query the database for the object class named groupOfNames:

  ```
  ContentEngine(config)# ldap server group static member-attribute member
  ContentEngine(config)# ldap server group static enable
  ```

- The following example shows how to query the database for the object class named groupOfUniqueNames:

  ```
  ContentEngine(config)# ldap server group static member-attribute uniquemember
  ContentEngine(config)# ldap server group static enable
  ```

**Note**

The ACNS 5.x software now supports static group queries of an LDAP user database. Static groups are supported on OpenLDAP servers, third-party LDAP servers, and Microsoft Active Directory servers.
Chapter 10 Configuring Content Authentication and Authorization on Standalone Content Engines

Configuring an Authentication Service on Standalone Content Engines

The following example shows how an LDAP server administrator can configure static groups in an LDAP database:

1. The LDAP server administrator defines the following nodes in the LDAP database:
   - The root node named .com
   - The organization node named cisco
   - A node for the organizational unit (ou) named Engineering
   - A node for the Hardware Engineers group that is nested under the Engineering group
   - A node for the Software Developers group that is nested under the Engineering group
   - An individual node for the user named Jay Doe
   - An individual node for the user named Don Smith

   The following example shows how the LDAP server administrator defines a node for the user Jay Doe:

   ```
   # Jay Doe, Engineers, cisco, com
   dn: cn=Jay Doe,ou=Engineering,dc=cisco,dc=com
   telephoneNumber: (408) 123-8910
   mail: jdoe8@cisco.com
   uid: jdoe8
   givenName: Jay
   sn: Doe
   cn: Jay Doe
   ```

   The following example shows how the LDAP server administrator defines a node for the user Don Smith:

   ```
   # Don Smith, Engineers, cisco, com
   dn: cn=Don Smith,ou=Engineering,dc=cisco,dc=com
   telephoneNumber: (408) 123-4567
   mail: dsmith7@cisco.com
   uid: dsmith7
   givenName: Don
   sn: Smith
   cn: Don Smith
   ```

2. The LDAP server administrator assigns specific members to static groups.

   The following example shows how the LDAP server administrator explicitly specifies that the parent group named Engineering has two static members (the groups named Hardware Engineers and Software Developers). The groups named Hardware Engineers and Software Developers are nested static groups; they are nested under the parent group.

   ```
   dn: cn=Engineering,dc=cisco,dc=com
   cn: Engineering
   objectclass: groupOfNames
   member: cn:Hardware Engineers,dc=cisco,dc=com
   member: cn:Software Developers,dc=cisco,dc=com
   ```

   The following example shows how the LDAP server administrator explicitly assigns Jay Doe and Don Smith to the static group named Hardware Engineers. This is a one-way link between the two connected nodes because the member attribute is used. If the member of attribute were used; a two-way link would be created between the two connected nodes. Connecting nodes with a two-way link reduces the number of TCP requests that are required to search through the database for user membership information.

   ```
   dn: cn=Engineers,dc=cisco,dc=com
   cn: Hardware Engineers
   object: groupOfNames
   ```
By default, static group queries and nested group queries are disabled on the Content Engine (as shown in the following excerpt of sample output from the `show ldap` EXEC command).

```
ContentEngine# show ldap
LDAP Configuration:
-------------------
Static Groups:           disabled
Group Attribute:
Group Member:           disabled
Nested Groups:        disabled
Nested Level:         1
```

However, you can use the CLI to enable and configure such queries on a standalone Content Engine, as described in these sections:

- Searching for User Account Information for LDAP Direct Static Groups, page 10-36
- Searching for User Account Information for LDAP Nested Static Groups, page 10-37

**Grouping Users into Active Directory Groups**

Microsoft Active Directory is a software application that runs on a Windows 2000 server. An Active Directory (AD) database is a user database that resides on a Windows 2000 server that is running the Microsoft Active Directory program.

In the ACNS 5.1 software and later releases, the LDAP client on a Content Engine provides LDAP support for Active Directory groups. Microsoft Active Directory only supports LDAP Version 3. By default, the Content Engine uses LDAP Version 2. Therefore, use the `ldap server version 3` global configuration command to configure the Content Engine to use LDAP Version 3 before enabling the LDAP Active Directory feature on the Content Engine.

```
ContentEngine(config)# ldap server version 3
```

To request that the LDAP server query the Active Directory group membership:

```
ContentEngine(config)# ldap server group active-directory enable
```

**Note**

The Active Directory group attribute is an LDAP Version 3 extension, and must be queried separately.

The following is an entry sample from a user account in an Active Directory database:

```
dn: CN=Penny Gold,CN=Users,DC=cisco,DC=local
memberOf: CN=Marketing,DC=cisco,DC=local
```

The `memberOf` attribute and a group name for each individual group were added to the user’s account configuration. The `memberOf` attribute does not support the nested group structure.

**About LDAP Directory Searches**

The LDAP directory service is a global directory service that allows LDAP clients (Content Engines) to access information that is stored in an LDAP directory, as follows:

1. The Content Engine connects to the specified LDAP server and queries the LDAP server for specific user membership information (for example, user identification [userid] and user password [userpassword]).
The following example shows how to configure the Content Engine to query a standard LDAP server for user identifications (user ID and password), and to ask the LDAP server to retrieve the names of any organizational units (groups) that a user belongs to:

```
ContentEngine(config)# ldap server userid-attribute uid
ContentEngine(config)# ldap server organizationUnit enable
```

The Content Engine administrator configures the search criteria on the Content Engine. For more information on this topic, see the “Querying LDAP Servers for User Membership Information” section on page 10-36.

2. The LDAP server searches its user database (an LDAP directory) for any entry that matches the specified search criteria.
   a. The LDAP server checks whether the `organizationUnit` option is enabled on the Content Engine:
      - If the `organizationUnit` option is enabled, the LDAP server queries the database for the organizationUnit configuration (`ou` attribute) of the user account. If the user belongs to more than one organizational unit, the LDAP server returns a string that contains all of the group names to which the user belongs (for example, Marketing, Engineering).
      - If the `organizationUnit` option is disabled, then the LDAP server does not perform the query.
   b. The LDAP server checks whether the Active Directory group (`memberOf`) option is enabled:
      - If the `memberOf` option is disabled, then the LDAP server (the Microsoft Active Directory server) does not perform the query.
      - If the `memberOf` option is enabled, then the LDAP server queries the Microsoft Active Directory server database for the Active Directory group configuration of the user account. The Microsoft Active Directory server collects the group names from the `memberOf` attribute of the user account, and returns this information to the Content Engine.
   c. The LDAP server checks whether the custom group option is enabled on the Content Engine:
      - If the custom group option is disabled, then the LDAP server does not perform this query.
      - If the custom group option is enabled, the LDAP server collects the group names from the custom attribute of the user account and returns this information to the Content Engine.
   d. The LDAP server checks whether the static group option is enabled on the Content Engine:
      - If the static group option is disabled, the LDAP server does not perform the query.
      - If the static group option is enabled, the LDAP server queries the database for the static group configuration. The server collects the names of the groups and returns this information to the Content Engine.

Note: The Active Directory group attribute is an LDAP Version 3 extension, and therefore must be queried separately.
3. The LDAP server responds to the query from the Content Engine with the requested information or with a message that it was unable to find the requested information:
   - If the user is not a valid user, then HTTP authentication fails. In this case, the Content Engine denies the user’s request to access the content.
   - If the user is a valid user, then HTTP authentication succeeds. If group-based authorization has not been enabled on the Content Engine, then the Content Engine grants the user access to the content at this point. If the Content Engine is configured for group-based authorization (for example, the access-lists 300 enable global configuration command has been used and the access lists have been defined), then the Content Engine uses the retrieved group names to determine whether the user should be granted access to the requested content.

   **Note**
   The retrieved user membership information is used for group-based authorization only. For more information about group-based authorization for LDAP users, see the “Example of Configuring LDAP Group Authorization with Group-Based Access Lists” section on page 10-47.

### Configuring the LDAP Cache on Standalone Content Engines

In the ACNS 5.4.1 software release, LDAP memory cache support for nested group searches was added. This feature enables the Content Engine to store the results of a nested group search locally in its LDAP cache. To support this new feature, the mem-cache option was added to the ntlm server ad-group-search global configuration command.

Use the ntlm server ad-group-search mem-cache global configuration command to enable and configure the LDAP cache on standalone Content Engines:

```plaintext
ContentEngine(config)# ntlm server ad-group-search mem-cache ?
enable   Enable ldap in-memory cache. (Default is Enabled)
max-ttl  Maximum amount of time from creation an entry is valid in the ldap in-memory cache
size     Maximum size of ldap in-memory cache to allocate in KBytes.
```

**Note**
The LDAP cache is sometimes called the “LDAP in-memory cache” because this cache resides on the Content Engine. “AD” is an abbreviation for Active Directory.

By default, the LDAP memory cache is enabled on the Content Engine, the cache size is set to 128 KB, and the maximum time to live (max-ttl) is set to 480 minutes. Valid values for the cache size are 128 to 10240 KB. Valid values for the maximum time to live are 1 to 1440 minutes.

To disable the LDAP memory cache feature, enter the no ntlm server ad-group-search mem-cache global configuration command.
Querying LDAP Servers for User Membership Information

This section provides some examples of how to configure the Content Engine to perform LDAP directory queries:

- Searching for User Account Information for Individuals in an LDAP Database, page 10-36
- Searching for User Account Information for LDAP Direct Static Groups, page 10-36
- Searching for User Account Information for LDAP Nested Static Groups, page 10-37

Note: All of these examples assume that the LDAP directory that is being queried has the structure depicted in Figure 10-3.

Searching for User Account Information for Individuals in an LDAP Database

The following example shows how to configure a Content Engine to search for user account information for individuals (for example, Jane Doe) in an LDAP user database:

Step 1 Specify the LDAP server that the Content Engine should use for this database search.

ContentEngine(config)# ldap server 172.16.1.1

By default, port 389 is configured as the TCP port for the LDAP authentication server. To specify another port for the LDAP server, use the `ldap server port port-num` command. Valid port numbers are 1 through 65535.

Step 2 Specify the base distinguished name that is the starting point for this database search.

ContentEngine(config)# ldap server base "dc=cisco,dc=com"

Step 3 Enable LDAP authentication on the Content Engine.

ContentEngine(config)# ldap server enable

Step 4 Specify the search criteria.

a. In this case, the LDAP server is requested to search for user IDs (username and user password).

ContentEngine(config)# ldap server userid-attribute uid

b. Enable the `organizationUnit` option on the Content Engine. This instructs the LDAP server to retrieve the group name from the `ou` attribute of the user account.

ContentEngine(config)# ldap server organizationUnit enable

Searching for User Account Information for LDAP Direct Static Groups

The following example shows how to configure the Content Engine to request that the specified LDAP server perform a direct (nonnested) static group database query. This type of query enables the Content Engine to perform HTTP request authentication for users who have been assigned to a direct static group (a parent static group).

In this example, the LDAP database is queried for user account information for the direct static group named Engineering.

Step 1 Specify the LDAP server that the Content Engine should use for this database search.
Step 2  Specify the group attribute to query.
In this example, the LDAP server will search the static group configurations for the group attribute
textual named cn for common name:
```
ContentEngine(config)# ldap server group static group-attribute cn
```

Step 3  Specify the group member attribute to query.
In this example, the LDAP server searches the static group configurations for the group member attribute
textual named member:
```
ContentEngine(config)# ldap server group static member-attribute member
```

Step 4  Specify the base distinguished name that is the starting point for this database search.
```
ContentEngine(config)# ldap server base "dc=cisco,dc=com"
```

Step 5  Enable LDAP authentication on the Content Engine.
```
ContentEngine(config)# ldap server enable
```

Step 6  Enable static group queries of an LDAP user database.
```
ContentEngine(config)# ldap server group static enable
```

### Searching for User Account Information for LDAP Nested Static Groups

The following example shows how to configure the Content Engine to request that the specified LDAP server perform a nested static group database query. In this case, the LDAP directory is searched for user account information for any static groups that are nested under the parent group named Engineering.

---

Step 1  Specify the LDAP server that the Content Engine should use for this database search.
```
ContentEngine(config)# ldap server 172.16.1.1
```

Step 2  Specify the nested level of the static group that you want to search for in the LDAP directory.
By default, the LDAP server searches the LDAP directory one level down from the starting point of the search. In this case, the two nested static groups Hardware Engineers and Software Developers are nested two levels down from the starting point of the search (the organizational node named cisco). In this example the LDAP server is instructed to search two levels down.
```
ContentEngine(config)# ldap server group static nested level 2
```

Step 3  Specify the group attribute to query.
In this example, the LDAP server searches the nested static group configurations for the group attribute
textual named cn for common name:
```
ContentEngine(config)# ldap server group static group-attribute cn
```

Step 4  Specify the group member attribute to query.
In this example, the LDAP server searches the static group configurations for the group member attribute
textual named member:
```
ContentEngine(config)# ldap server group static member-attribute member
```

Step 5  Specify the base distinguished name that is the starting point for this database search.
ContentEngine(config)# ldap server base "dc=cisco,dc=com"

**Step 6** Enable LDAP authentication on the Content Engine.
ContentEngine(config)# ldap server enable

**Step 7** Enable nested static group queries of the LDAP database.
ContentEngine(config)# ldap server group static nested enable

---

**Configuring the NTLM Authentication Service**

In the ACNS 5.4.1 software release, support of NTLM authentication for nontransparent FTP native requests was added.

In the ACNS 5.2.1 software and later releases, the following enhancements for NTLM HTTP request authentication are available:

- Support for up to eight NTLM servers for HTTP request authentication—Ability to configure the Content Engine to use up to eight NTLM servers for HTTP request authentication for load-balancing purposes. The ACNS 5.1.x software and earlier releases supported failover only. For more information, see the “About NTLM Load Balancing for HTTP Request Authentication” section on page 10-40.

- Support for up to eight Global Catalog servers for Active Directory group searches—Ability to configure the Content Engine to use up to eight Global Catalog servers for Active Directory group searches. See the “Configuring Content Engines for Active Directory Group Searches” section on page 10-49.

**Note** The order of server configuration determines the order of load balancing or failover. For example, if failover is enabled, then the first server configured (Server 1) is the primary server and is sent all of the requests first. The last server configured (Server 8) is the last server that the Content Engine contacts. If load balancing is enabled, only the first request is sent to the first configured server (Server 1), after which round-robin is used among the remaining servers (for example, the second request is sent to Server 2, and the third request is sent to Server 3).

- Changes to the Active Directory group search feature—LDAP queries are sent to the same Active Directory server that is assigned to perform the authentication unless the LDAP query fails, in which case the Content Engine sends the authorization request to the next configured server (the Content Engine only tries one more server).

- If the NTLM nested group search feature is enabled, you no longer need to configure the ldap-search-server host global configuration command. The Content Engine automatically uses the IP address of the configured NTLM server to send the LDAP queries.

- New `scheme` command option for NTLM servers—A `scheme` option was added to the `ntlm server` and `ntlm server ad-group-search gc-server` global configuration commands. This option allows you to specify the scheme (load balancing or failover) that is to be used among the configured NTLM or Global Catalog servers. The default scheme is failover. To specify the scheme for the NTLM servers for HTTP request authentication, use the `ntlm server scheme` global configuration command.
command. To change the scheme for the Global Catalog servers for Active Directory group searches, use the `ntlm server ad-group-search gc-server scheme` global configuration command as shown in the second example:

```
ContentEngine(config)# ntlm server ?
  ad-group-search     Active Directory group search options
  connection-retry    Maximum attempts to connect to server
  connection-timeout  Time to wait connecting to server (second
  domain              Specify Domain name
  enable              Enable NTLM Authentication
  host                Host options
  scheme              Scheme to use for the host list
ContentEngine(config)# ntlm server ad-group-search gc-server ?
  host    Specify global catalog server address
  port    Specify global catalog server port, default 3268
  scheme  Scheme to use for the host list
```

- **Polling thread**—Once one of the configured NTLM or Global Catalog servers is marked as dead, it is removed from the load-balancing or failover farm to prevent the Content Engine from directing incoming requests to it. The Content Engine periodically polls the dead server (every 30 seconds). If the Content Engine receives a response from the server, it adds the server back into the load-balancing or failover farm.

- **Authentication method controls for NTLM**—Ability to enable or disable the Content Engine from sending a basic authentication response header along with an NTLM authentication header. For more information, see the “Configuring the Authentication Method Control for NTLM HTTP Request Authentication” section on page 10-45.

- **Support for no default NTLM domain**—If the client does not supply a domain name in the request authentication credential and there is no default domain configured on the Content Engine, then an authentication error is returned to the client. A predetermined error page that contains text indicating the reason for the error is sent to the client. This feature is also referred to as the “no domain configuration” feature.

  **Note** The no domain configuration feature is only supported with browsers that do not support NTLM (for example, Netscape 7.1 or earlier browsers [Netscape 7.2 and later browsers support NTLM]). For the Netscape browser, the user must specify the domain if the Content Engine does not have an NTLM default domain configured; otherwise the client receives an error message. For the Netscape browser, the domain can only be supplied as part of the username in the format domain\username. Browsers that do support NTLM such as Internet Explorer, always include a domain name in the authentication credentials that originate from either the user being prompted to specify the credentials or from the domain that was used to log in the user on to the desktop.

- **Configurable allow domain list**—Ability to specify the list of domains that are allowed to perform NTLM HTTP request authentication with the Content Engine. For more information, see the “Configuring a List of Allowed Domains List NTLM HTTP Request Authentication” section on page 10-44.

- **To configure the Content Engine to send the username and domain name to the transaction log**, use the `transaction-logs log-window-domain` global configuration command. The Windows domain name that is used for NTLM authentication appears in the username field of the transaction log. The username appears in the format domain\username in those formats that contain usernames that are in Extended Squid-style or custom format using the `%u` format token.
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.

For clients outside the domain using the Netscape 7.1 or earlier browser, a dialog box appears and the first authentication request asks the client to enter a username and password. Once the client is successfully authenticated, the entry is placed in the cache, and no reauthentication requests are made to the client until the authentication cache entry expires.

- In the ACNS 5.3.x software and earlier releases, NTLMv1 is supported for NTLM HTTP request authentication (that is, NTLM request authentication for requests over the HTTP protocol). In the ACNS 5.4.1 software and later releases, both NTLMv1 and NTLMv2 are supported for NTLM HTTP request authentication. For more information on this topic, see the “About NTLM Support on Content Engines” section on page 10-3.

### About NTLM Load Balancing for HTTP Request Authentication

In the ACNS 4.x software to the ACNS 5.1.x software, you needed to configure one primary domain controller for HTTP request authentication and a secondary domain controller for failover. However, in large-scale networks, if all the traffic passes through the Content Engine, even though the Content Engine authentication cache can help reduce the load on the domain controller, it may still be impractical to have a single domain controller handle authentication queries from all of the end users.

To address this concern, load balancing between domain controllers was added in the ACNS 5.2 software. With the ACNS 5.2.1 software and later releases, you can configure up to eight servers (domain controllers) for load-balancing and failover purposes. The order of server configuration determines the order of load balancing or failover.

When load balancing is selected as the scheme, the requests are round-robined between the domain controllers. For example, if you have $n$ servers (domain controllers), the first request goes to Server 1; the second request is sent to Server 2, the $n$th request is sent to Server $n$, and the $(n+1)$th request is sent to Server 1. If Server 1 fails, the Content Engine attempts to send the request to the next configured server that is alive (in this case, Server 2). However, failover to the next alive server occurs only once. For example, if Server 2 goes down when handling request 1, then request 1 does not fail over again.

If load balancing is enabled and the server information is changed during run time, the change is picked up at run-time without disrupting the service. The configuration of each configured NTLM or Global Catalog server is available through the `show ntlm` EXEC command. Statistics about the total number of requests going through the servers is collected and available through the `show statistics ntlm` EXEC command. Statistics about requests going through each domain controller are also collected and available through the `show statistics ntlm` EXEC command.

---

**Note**

If the Active Directory nested group search is enabled, only servers in the same domain are supported. If the Active Directory nested group search is not enabled, servers in multiple domains are supported if the servers have a trusted relationship.

### Configuring the Content Engine to Use NTLM Servers for HTTP Request Authentication

You can use the Content Engine GUI or the CLI to configure a standalone Content Engine to use external NTLM servers for HTTP request authentication.
In the ACNS 5.1.x software and earlier releases, you explicitly designated a primary NTLM server and a secondary NTLM server by using the **primary** and **secondary** options of the `ntlm server host` global configuration command, as shown below:

```
ContentEngine(config)# ntlm server host 172.16.10.10 primary
ContentEngine(config)# ntlm server host 172.16.10.12 secondary
```

In the ACNS 5.2.1 software and later releases, you can configure a Content Engine to use up to eight NTLM servers for HTTP request authentication. The order of the server configuration determines the order of load balancing or failover. For example, if the failover is enabled then the first server configured (Server 1 that has an IP address of 172.16.10.10) is the primary server and is sent all of the requests first. The last server configured (Server 3 that has the IP address of 172.16.10.14) is the last server that the Content Engine contacts. If the load balancing is enabled, only the first request is sent to the first configured server (Server 1), after which round-robin is used among the remaining servers (for example, the second request is sent to Server 2, and the third request is sent to Server 3).

```
ContentEngine(config)# ntlm server host 172.16.10.10
ContentEngine(config)# ntlm server host 172.16.10.12
ContentEngine(config)# ntlm server host 172.16.10.14
```

**Note**
In the ACNS 5.2 software release, the `ntlm server host primary` option and the `ntlm server host secondary` options were removed because up to eight servers are now supported. In the ACNS 5.2.1 software release, the `ntlm server host scheme load-balanced` option was added.

You can use the Content Engine GUI or the CLI to configure the Content Engine to use up to eight NTLM servers for HTTP request authentication.

From the Content Engine GUI, choose **Caching > NTLM** to access the NTLM window. Use the NTLM window to specify NTLM server settings on the Content Engine, and click **Update**. For more information about the fields on the NTLM window, click the **HELP** button in the window.

From the Content Engine CLI, use the `ntlm server` global configuration command.

The following example describes how to use the Content Engine CLI to configure a standalone Content Engine to use the maximum number of servers (eight NTLM servers) to load balance HTTP authentication requests:

---

**Step 1**
Specify the hostname or IP address of each NTLM server that you want the Content Engine to use for HTTP request authentication by using the `ntlm server host` global configuration command. This list of configured NTLM servers is referred to as the *host list*.

In the following example, the Content Engine is configured to use a host list that consists of eight NTLM servers:

```
ContentEngine(config)# ntlm server host 172.16.10.10
ContentEngine(config)# ntlm server host 172.16.10.12
ContentEngine(config)# ntlm server host 172.16.10.14
ContentEngine(config)# ntlm server host 172.16.10.16
ContentEngine(config)# ntlm server host 172.16.10.18
ContentEngine(config)# ntlm server host 172.16.10.20
ContentEngine(config)# ntlm server host 172.16.10.22
ContentEngine(config)# ntlm server host 172.16.10.24
```

**Step 2**
Specify whether the Content Engine is to use NTLMv2 for request authentication when communicating with any of the configured NTLM servers.
By default, the Content Engine will use LM or NTLMv1. In the ACNS 5.4.1 software release, NTLMv2 support for request authentication of HTTP requests was added. By default, the NTLMv2 feature for request authentication is disabled on the Content Engine, and the Content Engine will use NTLMv1. In the following example, the Content Engine is configured to use NTLMv2 for request authentication with any of the configured NTLM servers:

```
ContentEngine(config)# ntlm version 2
```

For more information on this topic, see the “About NTLM v2 Support for Request Authentication” section on page 10-6.

**Step 3**
Specify the maximum number of times that the Content Engine is to attempt to connect to one of the configured NTLM servers.

In the following example, this value is set to 3:

```
ContentEngine(config)# ntlm server connection-retry 3
```

The default is two attempts. Valid values are from one to three attempts. After the specified number of attempts is exceeded, the Content Engine stops attempting to connect to the NTLM server and attempts to connect to the next configured server on the host list.

**Step 4**
Specify how long the Content Engine should wait for a response from the NTLM server that it is attempting to connect.

In the following example, this timeout is set to 10 seconds:

```
ContentEngine(config)# ntlm server connection-timeout 10
```

This is the timeout for one connection attempt. If the specified amount of time is exceeded, the Content Engine gives up the connection and attempts to connect to the same server up to the specified number of times (the number of retries specified with the `ntlm server connection-retry` global configuration command) before the Content Engine attempts to connect to the next server. The default is 5 seconds. Valid values are from 1 to 20 seconds.

**Step 5**
Specify whether the configured NTLM servers are to be used for failover or load balancing.

```
ContentEngine(config)# ntlm server scheme ?
fail-over      Fail-over between hosts
load-balanced Round-robin load balancing between hosts
```

By default, the configured servers are used for failover. In the following example, this default is changed, and the Content Engine is configured to use the configured servers for load balancing:

```
ContentEngine(config)# ntlm server scheme load-balanced
```

When load balancing is enabled, only the first request is sent to the first configured server, after which round-robin is used among the remaining configured servers. (In contrast, when failover is enabled, the Content Engine all the requests to the first configured server.)

**Step 6** (Optional). Use the `ntlm server ad-group-search mem-cache` global configuration command to configure the LDAP cache on the Content Engine that is running the ACNS 5.4.1 software and later releases.

By default, the LDAP cache is enabled on the Content Engine, the cache size is set to 128 KB, and the maximum time to live (max-ttl) is set to 480 minutes. The Content Engine uses its LDAP cache to store the results of a nested group search.

In the following example, the maximum size of the LDAP cache is set to 140 KB:

```
ContentEngine(config)# ntlm server ad-group-search mem-cache size 140
```

Warning: This config destroys and recreates the memcache with new size
For more information, see the “Configuring the LDAP Cache on Standalone Content Engines” section on page 10-35.
**Step 7**  Enable NTLM authentication on the Content Engine.

```
ContentEngine(config)# ntlm server enable
```

**Step 8**  View the NTLM configuration on the Content Engine.

```
ContentEngine# show ntlm
```

Check the command output to verify that the displayed NTLM configuration reflects the configuration that you just specified (for example, the specified NTLM servers are listed, and load balancing is specified). For an example of the `show ntlm` EXEC command output, see the “Displaying the Current NTLM Configuration for Standalone Content Engines” section on page 10-45.

---

**Configuring a List of Allowed Domains List NTLM HTTP Request Authentication**

In the ACNS 5.1.x software, you were required to specify the name of the Windows NT domain that the end user was to be authenticated against. This was referred to as the default NTLM domain name. For example, the following command specified that the user needed to be authenticated against the domain named “cisco.com” in order to be authenticated.

```
ContentEngine(config)# ntlm server domain-name cisco.com
```

In the ACNS 5.2.1 software and later releases, you are not required to specify a name for the default domain. If the client does not supply a domain name in the request authentication credential and there is no default domain configured on the Content Engine (that is, the `ntlm server domain` global configuration command was not used), then an authentication error message is returned to the client. A predetermined error page that contains text indicating the reason for the error is sent to the client.

In the ACNS 5.2.1 software and later releases, you can specify a list of domains that are allowed to perform NTLM HTTP request authentication with the Content Engine. This capability allows you to limit the domains that can perform NTLM HTTP request authentication with the Content Engine, which provides additional security. This feature is called the allowed domain feature. If this feature is enabled on the Content Engine and the supplied domain credential does not match any of the domains in the allowed domain list, then HTTP request authentication fails and the client is sent an error message.

To support the allowed domain feature, the following Content Engine CLI commands were added in the ACNS 5.2 software release:

- `ntlm allow-domain enable`—Enables the allowed domain list feature on the Content Engine. By default, the allow domain feature is disabled.
- `no ntlm allow-domain enable`—Disables the allowed domain list feature on the Content Engine.
- `ntlm allow-domain domain-name`—Define the names of the domains that are allowed to perform NTLM HTTP request authentication with the Content Engine. A domain list can contain up to 32 domain names.

If the allowed domain list feature is enabled, then this feature works as follows:

- If the client’s domain credential matches any domain in the configured domain list, the Content Engine performs NTLM HTTP request authentication for this content request. A case-insensitive comparison is used to check whether the specified domain is listed in the allowed domain list.
• If the client’s domain credential does not match any domain in the configured domain list or there are no domains configured on the allowed domain list, the Content Engine denies this content request and sends the client a 407 or 401 authentication error message. The 407 or 401 authentication message has a specific predetermined error page that contains text indicating the reason for the error.

**Configuring the Authentication Method Control for NTLM HTTP Request Authentication**

By default, the Content Engine (the HTTP proxy server) always sends a basic authentication response header along with an NTLM authentication header to the client browser. This default behavior enables the client to be authenticated with the Content Engine even if the client browser does not support the NTLM protocol, as is the case with the Netscape browser. (Internet Explorer supports the NTLM protocol.)

Because basic authentication transmits user credential information in clear text format, it is less secure than NTLM authentication. Consequently, for security purposes you may want to configure the Content Engine to not send a basic authentication response header along with an NTLM authentication header.

In the ACNS 5.2.1 software and later releases, you can configure the authentication method control for NTLM HTTP request authentication. The authentication method control feature allows you to enable or disable the Content Engine from sending a basic authentication response header along with an NTLM authentication header. To support this feature, the following Content Engine CLI commands were added in the ACNS 5.2.1 software release:

- **ntlm basic-auth enable**—Configures the Content Engine to send a basic authentication response header along with an NTLM authentication header to the client browser.
- **no ntlm basic-auth enable**—Configures the Content Engine to not send the basic authentication response header along with an NTLM authentication header, or to not honor it in a request.

If you do not want the client browser to be able to use the basic authentication method between the client and the Content Engine for NTLM HTTP request authentication because it is a less secure method than NTLM, then disable the NTLM basic authentication feature on a standalone Content Engine.

To disable the NTLM basic authentication feature on a standalone Content Engine, enter the **no ntlm basic-auth enable** global configuration command.

If the Content Engine is configured to not send the basic authentication header to the client and the client does not support NTLM authentication (for example, Netscape browsers only support basic authentication), then the client cannot continue with this HTTP request. The client browser behavior is browser-dependent; for example, some browsers may retry the request over a certain period of time.

**Displaying the Current NTLM Configuration for Standalone Content Engines**

To display information about each NTLM server and each Global Catalog server that the Content Engine is configured to use, enter the **show ntlm** EXEC command.

To display NTLM statistics such as the number of requests that were authenticated or denied, and a breakdown of statistics for each configured NTLM server and Global Catalog server, use the **show statistics ntlm** EXEC command.
**Configuring Group-Based Authorization for HTTP Requests**

For HTTP requests, the ACNS 5.x software supports group-based access lists for LDAP and NTLM users. In ACNS software releases prior to 5.1, group name-based access control lists were supported but not static groups. This group name-based access lists feature is called **group-based authorization**. By default, the access lists feature is disabled on a Content Engine. Group information will be checked and applied only if the access lists feature is enabled on the Content Engine.

*Note* In the ACNS 5.2.1 software release, group-based rules were also added and can be used for group-based authorization. For more information about groupname-based rules, see Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”

In Windows NT and Windows 2000 domains, administrators can use the Windows group feature to create groups in order to organize security principles, including user and other resources. An Active Directory (AD) database is a user database of a Windows 2000 server. This database can be queried for authentication purposes when LDAP or NTLM is used.

**Configuring Group-Based Access Lists on Standalone Content Engines**

To configure a standalone Content Engine to use access lists for group-based authorization, follow these steps:

**Step 1**  
Permit or deny a group from accessing the Internet using a standalone Content Engine by using the `access-lists 300` global configuration command.

The following example shows how to use access lists to permit access to groups within the base string named cisco based on organizational units such as Marketing or Engineering using the `access-lists 300 permit groupname` global configuration command. Group access is allowed for any user in the Marketing and Engineering groups. A user who does not belong to any of these groups is denied access with the `access-lists 300 deny groupname any` global configuration command.

```
ContentEngine(config)# access-lists 300 permit groupname Marketing
ContentEngine(config)# access-lists 300 permit groupname Engineering
ContentEngine(config)# access-lists 300 deny groupname any
```

For Windows-based user groups, you must append the domain name in front of the group name in the form domain\group:

- For Windows NT-based user groups, use the domain NetBIOS name.
- For Windows 2000-based user groups, use the domain DNS name.

*Note* From the Content Engine GUI, choose System > Access Lists, and use the displayed Access Lists window to define the entries for the access list. For more information about how to use the Access Lists window, click the HELP button in the window.

**Step 2**  
Enable the group name-based access list feature on the Content Engine.

```
ContentEngine(config)# access-lists enable
```
Chapter 10  Configuring Content Authentication and Authorization on Standalone Content Engines

Configuring an Authentication Service on Standalone Content Engines

Note  From the Content Engine GUI, choose System > Access Lists, and click the Enable access lists On radio button.

Example of Configuring LDAP Group Authorization with Group-Based Access Lists

In ACNS releases prior to 5.1, group name-based access lists were supported but not static groups. To ensure interoperability of the Content Engine group authentication support with the Microsoft Active Directory database, the ACNS 5.1 software and later releases support LDAP group-based authorization for static groups.

In this scenario, group access to the Internet is allowed to the following users:

- Any user who belongs to the Marketing organizational unit of the company named cisco.
- Any user who belongs to the organizational unit named Engineering except for those users who belong to the group named Hardware Engineers. Members of the nested group named Hardware Engineers will be denied access because they belong to a group that has been explicitly denied access.

Note  This scenario assumes the LDAP directory has the structure depicted in Figure 10-3.

This scenario assumes that the LDAP administrator has defined the group named Engineering as the parent of the Hardware Engineers and Software Developers groups in the LDAP directory, as follows:

dn: cn=Engineering,dc=cisco,dc=com
  cn: Engineering
  objectclass: groupOfNames
  member: cn:Hardware Engineers,dc=cisco,dc=com
  member: cn:Software Developers,dc=cisco,dc=com

dn: cd=Hardware Engineers,dc=cisco,dc=com
  cn: Hardware Engineers
  object: groupOfNames
  member: cn=Jay Doe,ou=Engineering,dc=cisco,dc=com
  member: cn=Don Smith,ou=Engineering,dc=cisco,dc=com

dn: cn=Software Developers,cd=cisco,cd=com
  cn: Software Developers
  objectclass: groupOfNames
  member: cn=John Gold,ou=Engineering,dc=cisco,dc=com
  member: cn=John Smith,ou=Engineering,dc=cisco,dc=com

To configure a standalone Content Engine for LDAP group-based authorization, follow these steps:

Step 1  Enable access to the LDAP server. Enter a hostname or an IP address for the LDAP server.

In the following example, the IP address of the LDAP server is used.

ContentEngine(config)# ldap server host 10.1.1.1

Step 2  Specify the base distinguished name (dn) as the starting point for the directory search. In this example, the strings cisco and com are used for the directory search.

ContentEngine(config)# ldap server base "dc=cisco,dc=com"
Step 3  Enable LDAP authentication on this Content Engine.

```
ContentEngine(config)# ldap server enable
```

Step 4  Define which groups are granted or denied access to content that is served by this Content Engine by using the `<access-lists 300 groupname>` global configuration command.

In the following example, group access is granted to any user who is not a member of the nested static group named Hardware Engineering:

```
ContentEngine(config)# access-lists 300 deny groupname Hardware Engineering
ContentEngine(config)# access-lists 300 permit groupname any
```

Step 5  Enable group name-based access lists on the Content Engine.

```
ContentEngine(config)# access-lists enable
```

---

**Example of Configuring NTLM Group Authorization with Group-Based Access Lists**

The following example shows how to use the Content Engine CLI to configure NTLM group authorization with group-based access lists. In this example, NTLM group access is granted to the Engineering and Marketing groups in the company named cisco.

To configure a standalone NTLM group authorization with access lists on a standalone Content Engine, follow these steps:

---

Step 1  Enable access to the NTLM server. Enter either a hostname or an IP address of the NTLM server by using the `<ntlm server host>` global configuration command.

**Note** In the ACNS 5.2.1 software and later releases, you can configure up to eight NTLM servers for HTTP request authentication.

In the following example, three NTLM servers are configured:

```
ContentEngine(config)# ntlm server host 172.16.10.10
ContentEngine(config)# ntlm server host 172.16.10.12
ContentEngine(config)# ntlm server host 172.16.10.14
```

The order of server configuration determines the order of load balancing or failover. If failover is enabled, the Content Engine sends all of its requests to the first configured server (Server 1, which has an IP address of 172.16.101.0). In contrast, if load balancing is enabled, only the first request is sent to Server 1, after which round-robin is used (for example, the second request is sent to Server 2 [the server with the IP address 172.16.10.12], and the third request is sent to Server 3 [the server with IP address 172.16.10.14]).

Step 2  Configure the Content Engine to use the list of configured NTLM servers for load balancing. The default scheme is failover.

```
ContentEngine(config)# ntlm server scheme load-balanced
```

Step 3  (Optional) Specify a default domain.

```
ContentEngine(config)# ntlm server domain cisco.com
```
In the ACNS 5.1.x software, you were required to specify a default domain. In the ACNS 5.2.1 software and later releases, you are not required to specify a default domain. If the client does not supply a domain name in the request authentication credential and there is no default domain configured on the Content Engine (the `ntlm server domain` global configuration command was not used), then an authentication error message is returned to the client. A predetermined error page that contains text indicating the reason for the error is sent to the client.

In the ACNS 5.2.1 software and later releases, you can also specify a list of domains that are allowed to perform NTLM HTTP request authentication with the Content Engine. For more information, see the “Configuring a List of Allowed Domains List NTLM HTTP Request Authentication” section on page 10-44.

**Step 4** Enable NTLM authentication on this Content Engine.

```
ContentEngine(config) # ntlm server enable
```

**Step 5** Permit access for groups within the base string cisco, based on organizational units such as Marketing and Engineering, using the `access-lists 300 permit groupname` global configuration command.

In the following example, group access is granted to any user in the Marketing and Engineering groups. A user who does not belong to either of these groups is denied access with the `access-lists 300 deny groupname any` global configuration command:

```
ContentEngine(config) # access-lists 300 permit groupname MY_DOMAIN\Marketing
ContentEngine(config) # access-lists 300 permit groupname MY_DOMAIN\Engineering
ContentEngine(config) # access-lists 300 deny groupname any
```

**Step 6** Enable group name-based access lists on this Content Engine.

```
ContentEngine(config) # access-lists enable
```

---

### Configuring Content Engines for Active Directory Group Searches

In the ACNS software releases prior to 5.1, the Content Engine only supported local groups within a global group for NTLM group-based authorization. To ensure interoperability of the Content Engine NTLM group authentication support with the Microsoft Active Directory database, the ACNS 5.1 software and later releases support static groups.

The ACNS 5.1 software and later releases can retrieve nested group names using an LDAP recursive search and apply all the group-based access lists configured for the nested groups. When you use nested groups with Active Directory servers, the policies configured for parent groups are automatically applied to members in subgroups.

**Note**

There are three kinds of groups in an Active Directory: universal, global, and domain local.

To perform a recursive query, an enumeration user’s credentials must be provided to query the primary domain controller for a complete list of group names. An enumeration user is an account defined on the Content Engine to allow the Content Engine to perform a search on an Active Directory server. This enumeration user needs to have read privileges throughout the whole directory.

In the ACNS 5.1.x software, group name-based access lists were the only feature that would trigger an Active Directory group search. In the ACNS 5.2.1 software and later releases, the following additional features can trigger an Active Directory group search for HTTP requests:

- If group-based rules are configured in the Rules Template (see Chapter 13, “Configuring the Rules Template on Standalone Content Engines”)

---

Cisco ACNS Software Configuration Guide for Locally Managed Deployments

OL-9140-01

10-49
If ICAP is configured to append the authenticated-group header (see Chapter 12, “Configuring ICAP on Standalone Content Engines”)

If the SmartFilter product is enabled on the Content Engine (see the “Configuring URL Filtering with SmartFilter Software” section on page 11-39)

In the ACNS 5.1.x software, you explicitly designated a primary and a secondary Global Catalog server for Active Directory group searches. When the primary Global Catalog server was not reachable, the Content Engine attempted to contact the secondary Global Catalog server.

In the ACNS 5.1.x software, you explicitly designated a primary and secondary server for an Active Directory group search and to obtain group information. You used the `ntlm server ad-group-search ldap-search-server host` global configuration command with the `primary` and `secondary` keywords to explicitly designate a primary and a secondary server, as shown here:

```
ContentEngine(config)# ntlm server host 172.16.10.10 primary
ContentEngine(config)# ntlm server host 172.16.10.12 secondary
```

However, in the case of Active Directory, because the domain controller supports both NTLM and LDAP, it is not necessary to configure the primary and secondary server for an Active Directory group search. In the ACNS 5.1.x software, the following two Content Engine CLI commands performed the same task (that is, they were configuring the same Active Directory domain controller):

```
ntlm server host
ntlm server ad-group-search ldap-search-server host
```

Consequently, the `ntlm server ad-group-search ldap-search-server host` global configuration command was removed in the ACNS 5.2.1 software release. Backward compatibility is supported. Configurations performed with the ACNS 5.1.x software that have this removed Content Engine CLI command are silently accepted but ignored in the back end.

In the ACNS 5.2.1 software and later releases, the Active Directory domain controllers (hosts) that are configured using the `ntlm server host` global configuration command are used for both authentication and authorization if the Active Directory nested group search feature is enabled.

In the ACNS 5.2.1 software and later releases, you can configure up to eight Global Catalog servers for Active Directory searches. (In the ACNS 5.2.1 software release, the `ntlm server ad-group-search gc-server host primary` option and the `ntlm server ad-group-search gc-server host secondary` options were removed.) The order of server configuration determines the order of load balancing or failover. If failover is enabled, then the Content Engine sends all of its Active Directory search requests to the first configured Global Catalog server (Server 1). In contrast, if load balancing is enabled, only the first request is sent to Server 1, after which round-robin is used (for example, the second request is sent to Server 2, and the third request is sent to Server 3).

To configure the settings for the Global Catalog servers that you want the Content Engine to use for Active Directory group searches, use the `ntlm server ad-group-search gc-server` global configuration command. For example, to specify the IP address or hostname of the Global Catalog server that the Content Engine is to use for Active Directory group searches, use the `ntlm server ad-group-search gc-server host host-IP-address or hostname` command.

To specify the host domain name (for example, “abc1.local”) for the configured Global Catalog server, use the `ntlm server ad-group-search gc-server host domain domain-name` global configuration command. In the following example, the Content Engine is configured to use the Global Catalog server that has the host domain name of abc1.local.

```
ContentEngine(config)# ntlm server ad-group-search gc-server host 10.77.157.213 domain abc1.local
```
In the ACNS 5.2.1 software and later releases, you can use the `ldap-search-port` option of the `ntlm server ad-group-search` global configuration command to specify the LDAP port for group information retrieval. The default is port 389. This option configures the LDAP search server port for all of the configured Active Directory domain controllers.

**Note** The `ldap-search-port` option replaces the ACNS 5.1.x software `ldap-search-server port` option.

In the ACNS 5.2.1 software and later releases, you can use the `scheme` option of the `ntlm server ad-group-search gc-server` global configuration command to specify whether the configured Global Catalog servers are to be used for load balancing or failover.

```bash
ContentEngine(config)# ntlm server ad-group-search gc-server ?
  host    Specify global catalog server address
  port    Specify global catalog server port, default 3268
  scheme  Scheme to use for the host list
```

In the ACNS 5.2.1 software and later releases, the NTLM load balancing feature is supported, which makes cross-domain authorization a more common deployment scenario. In the ACNS 5.3.1 software and later releases, support for the LDAP referral feature is available.

Support of LDAP referral enables the ACNS software to retrieve authorization information for a user who does not belong to the same domain as the configured Active Directory domain controller, but does belong to a trusted domain. When the Active Directory domain controller receives an LDAP query for a user who is not in its own domain, but is in a trusted domain, it will send back an LDAP referral URL to the Content Engine. If the LDAP referral feature is enabled on a Content Engine, the Content Engine will retrieve the information about the referred server in the referral URL, and contact the server to request the user’s authorization information.

Support for the LDAP referral feature provides the following capabilities:

- Support of Active Directory trusted domain user authorization
- Support of LDAP referral for NTLM nested group searches
- The ability to configure the LDAP nesting referral level
- The ability to configure Active Directory domain controllers from multiple domains for NTLM load balancing purposes

**Note** The ability to configure Active Directory domain controllers from multiple domains requires that the multiple domains are in a trusted relationship. Authentication and authorization will not be performed correctly if the multiple domain controllers from different nontrusted domains are configured.

In the ACNS 5.3.1 software and later releases, you can configure the LDAP referral feature through the `ntlm server ad-group-search ldap-referral` global configuration command:

```bash
ContentEngine(config)# ntlm server ad-group-search ldap-referral ?
  enable  Enable ldap referral. (Default is Disabled)
  limit   Maximum depth of nested referral to follow
```

By default, the LDAP referral feature is disabled on the Content Engine. To enable this feature, enter the `ntlm server ad-group-search ldap-referral enable` global configuration command. After enabling the LDAP referral feature on the Content Engine, you can disable it at a later time by entering the `no ntlm server ad-group-search ldap-referral enable` command.
To specify the referral limit for NTLM nested group searches, use the `ldap-referral limit` option of the `ntlm server ad-group-search ldap-referral` command. By default, five nested referrals are allowed for an NTLM nested group search. Valid values are from 1 to 10.

Even though the results of a first-level search can contain the desired results (the results that the Content Engine is searching for), Active Directory servers tend to return multiple nested referral URLs, which causes additional, unnecessary round trips to the Active Directory server. Consequently, you can reduce the referral limit to a smaller number if you are sure that the first few level search responses will contain the desired search result because of your directory structure.

For example, if the search result is contained in the first-level search response, you can configure the referral limit to 1 for performance purposes. By setting the referral limit to 1, the Content Engine will only follow one referral URL to contact the correct domain controller (Domain Controller A), and will not follow the additional, unnecessary referral URLs that are generated from Domain Controller A along with the search result.

The following example shows how to specify that 1 instead of 5 nested referrals are allowed for NTLM nested group searches:

```
ContentEngine(config)# ntlm server ad-group-search ldap-referral limit 1
```

In the ACNS 5.4.1 software release, support for caching the results of a nested group search locally in the LDAP cache on the Content Engine was added. To support this new feature, the `mem-cache` option was added to the `ntlm server ad-group-search` global configuration command:

```
ContentEngine(config)# ntlm server ad-group-search mem-cache ?
  enable   Enable ldap in-memory cache. (Default is Enabled)
  max-ttl  Maximum amount of time from creation an entry is valid in the ldap
           in-memory cache
  size     Maximum size of ldap in-memory cache to allocate in KBytes.
```

By default, the LDAP memory cache is enabled on the Content Engine, the cache size is set to 128 KB, and the maximum time to live (TTL) is set to 480 minutes.

To display the currently configured NTLM parameters on the Content Engine, enter the `show ntlm` EXEC command. The command output includes such information as whether the LDAP referral feature is enabled (for example, the command output shows “AD LDAP referral chasing: Enabled”), as well as the current referral limit (for example, the command output shows “AD LDAP referral chasing limit: 8”).

In the ACNS 5.4.1 software and later releases, the command output also includes configuration information about the local LDAP cache on the Content Engine (for example, if the LDAP cache is enabled as well as the maximum cache size and maximum time for an object to be stored in the cache [the maximum time to live [max-ttl] value]).

### Examples of Configuring Content Engines to Support Active Directory Group Searches and LDAP Caching

The following example shows how to use the `ntlm server ad-group-search` global configuration command to configure the Content Engine to support Active Directory group searches. In this example, the Content Engine is configured to use the Global Catalog server that has the IP address 10.77.157.213 for the Active Directory group search. The host domain name for this Global Catalog server is `abc1.local`. Load balancing is specified as the scheme (the default scheme is failover). Port 111 is specified as the LDAP port for group information retrieval.
By default, the NTLM LDAP memory cache feature is enabled on the Content Engine. Use the `ntlm server ad-group-search mem-cache max-ttl` global configuration command to change the maximum time for an object to live in the LDAP cache to 400 minutes (the default is 480 minutes). Next, use the `ntlm server ad-group-search mem-cache size` global configuration command to change the maximum size of the LDAP cache to 140 KB (the default is 128 KB) on the Content Engine.

After you have configured the Active Directory group search parameters and the LDAP cache, you must enter the `ntlm server ad-group-search enable` global configuration command to enable the Active Directory group search feature on the Content Engine:

```
ContentEngine(config)# ntlm server ad-group-search enum-user username administrator
ContentEngine(config)# ntlm server ad-group-search enum-user password ***
ContentEngine(config)# ntlm server ad-group-search enum-user domain abc1.local
ContentEngine(config)# ntlm server ad-group-search gc-server host 10.77.157.213 domain abc1.local
ContentEngine(config)# ntlm server ad-group-search gc-server host 10.77.157.214 domain abc1.local
ContentEngine(config)# ntlm server ad-group-search gc-server scheme load-balanced
ContentEngine(config)# ntlm server ad-group-search ldap-search-port 111
ContentEngine(config)# ntlm server ad-group-search mem-cache max-ttl 400
Warning: This config destroys and recreates the memcache with new ttl
ContentEngine(config)# ntlm server ad-group-search mem-cache size 140
Warning: This config destroys and recreates the memcache with new size
ContentEngine(config)# ntlm server ad-group-search mem-cache enable
ContentEngine(config)# ntlm server ad-group-search enable
ContentEngine(config)# ntlm server ad-group-search ldap-referral limit 1
```

When you enable Active Directory search groups, the access list must be configured with the correct domain name. The group name should look like this:

```
DNS domain name\group name
```

The following example uses access lists to enable Active Directory search groups:

```
ContentEngine(config)# access-lists 300 permit groupname mydomain.local\unix11_sec
ContentEngine(config)# access-lists 300 deny groupname any
ContentEngine(config)# access-lists enable
```

The LDAP queries are sent to the same Active Directory server that is assigned to perform authentication unless the LDAP query fails. If the LDAP query fails, the authorization request fails over to the next configured server. If the NTLM service or the LDAP service on the Active Directory server is not accessible, the Content Engine considers the Active Directory server nonfunctional.

### Disabling Group Name-Based Access Lists

You can disable the group name-based access list feature on a Content Engine without losing any of the configured access lists, as follows:

- From the Content Engine GUI, choose System > Access Lists. Click the Enable access lists Off radio button in the Access Lists window, and then click Update.
- From the Content Engine CLI, use the `no access-lists` global configuration command.
Configuring the LDAP Acceptable Use Policy Feature

The ACNS 5.1.1 software and later releases support the LDAP acceptable use policy feature. When a user opens a browser session, the Content Engine queries a specific LDAP attribute to determine whether the user has viewed and accepted the acceptable use policy (AUP). If a user has not accepted this policy, then the Content Engine redirects the user to an internal web page with the AUP, which the user must read and accept before being allowed to browse content through the Content Engine. Once the user accepts the policy, the Content Engine sets an LDAP attribute that allows the user full access to browse through the proxy server (the Content Engine).

This LDAP attribute is configurable and is an integer stored in the user’s database that represents the version of the policy that the user has accepted. This value is compared against the current version set in the Content Engine. If these values are equal, the user is given access to browse; otherwise, the user is redirected to the configured URL that sends the user to the internal web page that allows the user to read and accept the AUP.

To enable the AUP, enter the `ldap server policy-redirect enable` global configuration command. To specify the URL to which the user is redirected to view and accept the policy, use the `ldap server policy-redirect redirect-url url` global configuration command. To define the LDAP attribute that is to be queried for the version that the user has accepted, use the `ldap server policy-redirect attribute attribute` command.

To configure the AUP on a standalone Content Engine, follow these steps:

**Step 1**
Enable the AUP on the Content Engine.

```
ContentEngine(config)# ldap server policy-redirect enable
```

**Step 2**
Specify the URL that the user is redirected to in order to view and accept the policy.

```
ContentEngine(config)# ldap server policy-redirect redirect-url url
```

**Step 3**
Specify the LDAP attribute that the Content Engine should query to determine the version of the AUP that the user has accepted.

```
ContentEngine(config)# ldap server policy-redirect attribute aup-attribute
```

**Step 4**
Specify the current version of the AUP on the Content Engine. This value is a global value that is used for all users to view so that they can determine whether they have accepted this version of the AUP.

```
ContentEngine(config)# ldap server policy-redirect value latest-policy-version
```

Configuring the LDAP Password Expiration Feature

The LDAP password feature allows you to configure the Content Engine to redirect users to a web page, which will prompt them for a username and password. To configure the LDAP authorization password expiration feature, use the `ldap server password-expiry` global configuration command.
Configuring Request Authentication for Nontransparent FTP Native Requests

In the ACNS 5.4.1 software release, proxy authentication (that is, request authentication at the Content Engine), was added for nontransparent FTP native requests. With request authentication and authorization, the Content Engine is verifying the end user (for example, FTP clients such as a Reflection X client, WS-FTP clients, and UNIX or DOS command line FTP programs) that sent the request to the Content Engine. In contrast, end-to-end authentication and caching of authenticated objects deals with authentication for a particular object, and it is the origin server and not the Content Engine that verifies the end user.

In the ACNS 5.4.1 software release, the authentication option was added to the ftp-native proxy configuration command to support proxy authentication for nontransparent FTP native requests:

```
ContentEngine(config)# ftp-native proxy ?
    active-mode     Configuration of active mode for native ftp proxy
    authentication  Configuration for proxy authentication of proxy-mode requests
    incoming        Configuration for incoming proxy-mode requests
ContentEngine(config)# ftp-native proxy authentication ?
    enable  If an authentication service is configured then use it for
             authenticating proxy-mode ftp-native requests.
```

By default, the ftp-native proxy authentication feature is disabled. You can enable the ftp-native proxy authentication feature, as follows:

```
ContentEngine(config)# ftp-native proxy authentication enable
```

**Note**

Because the FTP protocol is inherently insecure, the authentication credentials can be sniffed off the network, which expose user credentials that otherwise would have been provided over a secure channel (for example, in the case of HTTP).

In the ACNS 5.4.1 software release, proxy authentication (that is, request authentication at the Content Engine) was added for nontransparent FTP native requests. With request authentication and authorization, the Content Engine is verifying the end user (for example, FTP clients such as a Reflection X client, WS-FTP clients, and UNIX or DOS command line FTP programs) that sent the request to the Content Engine. In contrast, end-to-end authentication and caching of authenticated objects deals with authentication for a particular object, and it is the origin server and not the Content Engine that verifies the end user.

In the ACNS 5.4.1 software release, the authentication option was added to the ftp-native proxy configuration command to support proxy authentication for nontransparent FTP native requests:

```
ContentEngine(config)# ftp-native proxy ?
    active-mode     Configuration of active mode for native ftp proxy
    authentication  Configuration for proxy authentication of proxy-mode requests
    incoming        Configuration for incoming proxy-mode requests
ContentEngine(config)# ftp-native proxy authentication ?
    enable  If an authentication service is configured then use it for
             authenticating proxy-mode ftp-native requests.
```

By default, the ftp-native proxy authentication feature is disabled. You can enable the ftp-native proxy authentication feature, as follows:

```
ContentEngine(config)# ftp-native proxy authentication enable
```
Because the FTP protocol is inherently insecure, the authentication credentials can be sniffed off the network, which expose user credentials that otherwise would have been provided over a secure channel (for example, in the case of HTTP).

Nontransparent proxy mode with proxy-authentication support requires that the FTP client provide the FTP proxy (that is, the Content Engine that is acting as the FTP proxy for the FTP client) the proxy username, (optionally) the proxy password, and the hostname of the origin FTP server.

When FTP clients and proxies are configured for nontransparent proxying with proxy authentication, the FTP client (TCP) connects to the FTP proxy (that is, the Content Engine) and issues FTP commands to authenticate with the proxy using one of the following methods:

- **FTP Proxy Authentication Method # 1**
  The client uses the FTP USER command to specify the server username and the hostname of the origin FTP server:
  ```
  USER proxy-username
  PASS proxy-password
  USER server-username@server-hostname.company.com
  PASS server-password
  ```

- **FTP Proxy Authentication Method # 2:**
  The client uses the FTP SITE command to specify the hostname of the origin FTP server:
  ```
  USER proxy-username
  PASS proxy-password
  SITE server-hostname.company.com
  USER server-username
  PASS server-password
  ```

The following is an example of an FTP client session when TACACS+ proxy authentication is enabled on the Content Engine. In this example, the FTP client uses the FTP proxy authentication method # 2 (for example, the FTP client enters the FTP SITE command to specify the hostname of the origin FTP server):

```
shell# ftp -d 2.9.192.11 8021
Connected to 2.9.192.11
220 Welcome to FTP-proxy.  Login to the proxy using username and password.
Name (2.9.192.11:tuser): tuser
---> USER tuser
331 Password required for tuser.
Password:
---> PASS XXXX
220 Welcome to FTP-proxy.
220 Login to origin server using the 'USER username@server-hostname' command, or
220 Login to origin server using the 'SITE server-hostname' followed by the 'USER
username' command.
ftp> site host.abccorp.com
---> SITE host.abccorp.com
ftp> user anonymous
---> USER anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
---> PASS XXXX
230 Guest login ok, access restrictions apply.
ftp> quit
```
If the proxy authentication succeeds for a particular client, the username that is provided by that client during the FTP proxy authentication process is logged in the transaction log if one of the following transaction logging formats have been configured on the Content Engine:

- Extended-squid logging
- Custom logging

**Note**

For the Custom transaction logging format, you must include the `%u` format-specifier when you configure the `transaction-logs format custom` command.

If the proxy authentication fails for an FTP client, the authentication failures are logged in the system log. Content Engine administrators can check the system log to monitor such authentication failures. For more information about enabling and configuring transaction logging on a standalone Content Engine, see the “Enabling Transaction Logging” section on page 21-33.

When configuring request authentication for nontransparent FTP native requests on a standalone Content Engine, remember the following important points:

- By default, the ftp-native proxy authentication feature is disabled on the Content Engine. Because the FTP protocol is inherently insecure, the authentication credentials can be sniffed off the network, which expose user credentials that otherwise would have been provided over a secure channel (for example, in the case of HTTP). In the ACNS 5.4.1 software and later releases, you can enable the ftp-native proxy authentication feature, as follows:

```
ContentEngine(config)# ftp-native proxy authentication enable
```

- If you enter the `ftp-native proxy authentication enable` command and you have not already configured an authentication service (that is, RADIUS, TACACS+, NTLM, or LDAP) on the Content Engine, a warning message will be displayed. This message indicates that you must configure an authentication service on the Content Engine before you can enable the FTP proxy authentication feature on the Content Engine.

- For request authentication for nontransparent FTP native requests, ACNS supports TACACS+, RADIUS, NTLM, and LDAP as an authentication service. Whether or not the FTP client, who has sent the FTP native request, is queried for proxy authentication by the Content Engine is based on whether one of the supported authentication services (RADIUS, TACACS+, NTLM, or LDAP) is enabled on the Content Engine that receives the incoming FTP native request.

In the ACNS 5.4.1 software and later releases, you can use the same process to enable and configure an authentication service for HTTP requests and nontransparent FTP native requests (for example, you use the same process to enable and configure RADIUS as an authentication service for HTTP requests and nontransparent FTP native requests). However, the following restrictions apply to FTP native caching support:

- No support for FTP request proxy rules
- No support for any URL filtering schemes (good list, bad list, N2H2, Websense, and SmartFilter)
For more information about configuring an authentication service to control request authentication on a standalone Content Engine, see the following sections:

- Configuring the RADIUS Authentication Service, page 10-19
- Configuring the TACACS+ Authentication Service, page 10-20
- Configuring the LDAP Authentication Service, page 10-21
- Configuring the NTLM Authentication Service, page 10-38

- In ACNS 5.4.1 software and later releases, you can use IP ACLs to control access to the native FTP proxy service that is running on the Content Engine. For more information, see the “About IP ACL Support for FTP Native Requests” section on page 7-42.

- In the ACNS 5.4.1 software and later releases, you can use the ftp-native custom-message global configuration command to configure customized response messages, which the Content Engine sends to an FTP client in response to an incoming proxy mode connection:

  
  ContentEngine# ftp-native custom-message 
  download  Download the custom message file specified by the URL to the CE
  reset     Revert to default message and delete the local file on the CE
  upload    Upload the custom message file to the specified host, directory and 
             filename using the FTP protocol

You can use the ftp-native custom-message EXEC command to create, upload, and download files that contain the following custom messages:

- A custom welcome message for welcoming proxy mode connections from FTP clients
- A custom error message when an FTP client is denied access based on an IP ACL that has been configured on the Content Engine for incoming FTP native requests.

For more information about this topic, see the “Creating Custom Messages for FTP Proxy Responses for FTP Native Requests” section on page 5-19.
Configuring Content Preloading and URL Filtering on Standalone Content Engines

This chapter provides an overview of content preloading and the different types of URL filtering that are supported with standalone Content Engines that are running the ACNS 5.4.x software or later releases, and describes how to configure content preloading and URL filtering on standalone Content Engines.

This chapter contains the following sections:

- Configuring Content Preloading for Standalone Content Engines, page 11-2
- Configuring URL Filtering on Standalone Content Engines, page 11-11
- Configuring Content Engines to Bypass URL Filtering for Specific HTTP and HTTPS Requests, page 11-41
- Displaying the Current URL Filtering Configurations, page 11-41
- Displaying URL Filtering Statistics, page 11-41
- Clearing URL Filtering Statistics, page 11-42

In the ACNS 5.2.3 software and later releases, you can configure a Content Engine to monitor the performance of specific URLs. The Content Engine maintains statistics about the various response characteristics for each of the monitored URLs. For more information on this topic, see the “Monitoring Critical Disk Drives on Standalone Content Engines” section on page 21-17.

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication.

For information about how to configure URL filtering on Content Engines that are registered with a Content Distribution Manager (as opposed to standalone Content Engines), see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Configuring Content Preloading for Standalone Content Engines

This section provides an overview of content preloading for standalone Content Engines that are running the ACNS 5.4.1 software and later releases. It also provides instructions on how to configure this feature on a standalone Content Engine.

Preloaded content is content that is retrieved and stored on a standalone Content Engine because the administrator of that Content Engine scheduled a retrieval of specific content in anticipation of user requests for that content. Content preloading is done by configuring the standalone Content Engine to create a cache request for all the content located at the origin web server that stores the primary content.

You can specify bandwidth limits for the preload process to ensure that bandwidth consumption does not exceed the specified bandwidth limits during the preload process. During the preload process, the Content Engine scans websites several link levels down for content, retrieves the specified content, and stores it locally for future requests. At a specified time, the Content Engine scans several levels of websites to verify that its content is still current, and it updates any content that has changed.

Note

The ACNS 5.x software can read a file of URLs and preload the specified URL content on the standalone Content Engine. The following type of content can be preloaded on a standalone Content Engine: HTTP URLs and FTP-over-HTTP URLs. All configured HTTP and FTP-over-HTTP parameters and rules apply to the preloaded objects.

Support of Preloading of NTLM Authenticated Objects

In the ACNS 5.1.1 software and later releases, the preloading of NTLM authenticated objects is supported. This feature allows NTLM authenticated objects (authenticated objects that reside on the servers that authenticate NTLM only) to be preloaded on a Content Engine.

Note

In the ACNS 5.1.1 software and later releases, NTLMv1 support of preloading NTLM authenticated objects on standalone Content Engines is available. In the ACNS 5.4.1 software release, NTLMv2 support of preloading NTLM authenticated objects was added. The ntlm version global configuration command, which was added in the ACNS 5.4.1 software release, has two command options: the version 1 option to re-enable NTLMv1, and the version 2 option to enable NTLMv2 on a standalone Content Engine. The version 1 command option is the default option.

An entry in a URL list file has the following format:

URL [depth] [domain-name: host-name: host-domain-name]

hostname and host-domain-name can be null; however, the domain name is required if NTLM credentials have been configured. (The separator is required.)

http://www.cisco.com 3 apac::

If NTLM-related information is not present in the preload URL list file entry, the authentication scheme falls back to basic authentication.

By default, the Content Engine does not cache basic and NTLM authenticated objects. To enable a standalone Content Engine to fetch specific objects and cache these objects that are authenticated with any authentication scheme (basic authentication or NTLM authentication), enter the http cache-authenticated all global configuration command.

ContentEngine(config)# http cache-authenticated all
To configure the Content Engine to cache only NTLM authenticated objects, enter the
**http cache-authenticated ntlm** global configuration command. The cached objects are tagged as
NTLM protected so that subsequent requests for these same objects are subjected to authentication
before the Content Engine can serve the content to the client.

Before you preload WMT streaming media files on the Content Engine, you must enable the WMT
feature on your Content Engine. If you used the Setup utility to configure WMT caching (as described
in Step 15 of the “Using the Setup Utility to Configure a Basic Configuration on a Standalone
Content Engine”) on your Content Engine, then WMT is already enabled on the Content Engine.
Otherwise, see the “Configuring WMT RTSP Streaming and Caching Services on Standalone
Content Engines” section on page 9-14 for instructions on how to use the Content Engine CLI (instead
of the Setup utility) to enable Windows Media services on a standalone Content Engine before you
enable the preloading of Windows Media streaming files for this Content Engine.

### Creating a Preload URL List File

The preload URL list file lists the URLs (HTTP or FTP-over-HTTP) to be preloaded on the
Content Engine. This file is maintained by the administrator and must be created on a remote system.
This file can be transferred to the standalone Content Engine for preloading access, or accessed from the
remote server. The **pre-load url-list-file path** global configuration command specifies the path of this
file.

**Note**

In the **pre-load url-list-file path** global configuration command, the value for **path** can be a URL or a
local file path.

You can place the list of URLs in a file on a local disk. You can also use the **mkdir** EXEC command to
make a subdirectory that contains the preload URL list file. For instance, the **mkdir**
/local1/preload-directory command creates a subdirectory called preload-directory on local disk
/local1.

Each URL in the preload URL list file has 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 preload depth level default of 3 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>
```

If you want to preload authenticated content to a Content Engine, the URL list file entry must be written
as follows:

```
http://username:password@www.authenticatedsite.com/ depth level
```

**Note**

In the ACNS 5.4.1 software and later releases, support for proxy authentication for preloaded content
was added. For more information on this topic, see the “Proxy Authentication Support for Content
Preloading” section on page 11-8.
When configuring a preload URL list file through the Content Engine CLI, the `pre-load url-list-file` global configuration command only had the HTTP or FTP option in the ACNS 5.1.x software earlier than the ACNS 5.1.5 software release. There was no method in place to fetch the preload URL list file securely.

In the ACNS 5.1.5 software release, the ability to fetch the preload URL file over HTTPS was added. If a preload URL list file contains usernames and passwords, organizations are now able to fetch the preload URL list file over HTTPS. The actual preloading of HTTPS links is not supported; only the downloading of the preload URL list file through the HTTPS protocol.

### Enabling and Configuring Content Preloading on Standalone Content Engines

You can enable and configure content preloading on a standalone Content Engine from either the Content Engine GUI or the CLI.

**Note**

From the Content Engine GUI, choose **Caching > Content Preload**. Use the displayed Content Preload window to enable and configure this feature on this standalone Content Engine. For more information about how to use the Content Preload window to perform this task, click the **HELP** button in the window.

To use the Content Engine CLI to enable and configure content preloading on a standalone Content Engine, follow these steps:

1. **Enable content preloading on the Content Engine.**
   ```
   ContentEngine(config)# pre-load enable
   ```
2. **Create the preload URL list file, as described in the “Creating a Preload URL List File” section on page 11-3.**
3. **Specify the maximum number of concurrent requests for the URL retrieval.** You can specify a value from 1 to 30 (for example, 24). The default is 10. If the number of URLs in the preload URL list file is fewer than the number of specified concurrent requests, then the lesser number is active.
   ```
   ContentEngine(config)# pre-load concurrent-requests 24
   ```
4. **Specify the default depth level for the URL retrieval (for example, four levels deep).** You can specify a value from 0 to 20. The default is 3. Setting the depth level default to 0 would be useful if you have specified URLs in preload.txt files and you do not want the Content Engine to try to preload other URLs.
   ```
   ContentEngine(config)# pre-load depth-level-default 4
   ```
Step 5  Specify the path of the file that contains the URL list or a URL.

ContentEngine(config)# pre-load url-list-file path

path is the path of the file that contains the URL list or a URL. For example:

pre-load url-list-file /local1/myurllist
pre-load url-list-file ftp://ftpserver/ftpdirectory/urllist.txt
pre-load url-list-file http://server/directory/urllist.txt

Note  The actual preloading of HTTPS links is not supported; only the downloading of the preload URL list file through the HTTPS protocol.

Step 6  Specify the domains to be fetched during the preload process (for example, cisco.com).

ContentEngine(config)# pre-load fetch domain cisco.com

Step 7  Specify that other domains in an HTML page should be traversed. By default, other domains in an HTML page are not traversed during the content preload.

ContentEngine(config)# pre-load traverse-other-domains

Step 8  Specify the suffixes to be excluded from the preload operation. This creates a filter for the objects that are to be excluded.

ContentEngine(config)# pre-load no-fetch suffix .mil .su .ca

Step 9  Configure a maximum bandwidth for the preloading process (for example, 50,000 kbps).

ContentEngine(config)# pre-load max-bandwidth 50000

Note  With the ACNS 5.x software, you can preload WMT streaming media files that may have different bit rates at the URL specified for content preloading. You can also control WMT bandwidth using the bandwidth wmt outgoing and bandwidth incoming global configuration commands. For more information, see the “Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates” section on page 9-23.

Step 10 To trigger a content preload immediately, enter the pre-load force EXEC command.

Step 11 To configure the Content Engine to preload specific content for a future time, use the pre-load schedule global configuration command. The Content Engine accesses the specified preload URL list file with a frequency set by the specified preloading schedule (set through the pre-load schedule global configuration command or the Content Engine GUI [Caching > Content Preloading]).

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. If you want to change this default, do the following:

a. To specify the start and end times for daily or weekly preloads, use hh:mm (where hh is the hour, and mm is the minutes, for example, 01:00). For hourly preloads, use mm to specify the start and end times. The following example shows how to specify a daily interval for scheduling the content preload. In this example, the preload operation starts every day at 1:00 AM and ends every day at 2:00 AM:

ContentEngine(config)# pre-load schedule every-day start-time 01:00 end-time 02:00
b. To specify the start time and end times for hourly preloads, the start time should be 0 and the end time should be 59. For daily and weekly preloads, the start time should be from 0 to 23, and the end time should be from 0 to 59. If the endtime option is not specified, the preload operation will continue until completion.

To configure a preload on more than one day of the week, use the `pre-load schedule every-week` global configuration command. The following example shows how to schedule a preload operation every week on Sunday and Wednesday from 1:00 AM to 6:00 AM:

```
ContentEngine(config)# pre-load schedule every-week Sun Wed start-time 01:00 end-time 06:00
```

**Step 12** Set the Type of Service (ToS) value as well as the differentiated services code point (DSCP) for all preload traffic by using the `pre-load dscp` global configuration command to set the Type of Service (ToS) value as well as the differentiated services code point (DSCP) for all preload traffic.

Setting the ToS or 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.

The ACNS 5.x software includes ToS or DSCP support for HTTP, FTP, and WMT preload traffic. Because content preloading is initiated by the Content Engine and not by the requesting client when a connection is made to an origin server, the ToS or DSCP code point on the traffic going toward the server needs to be set before contact is made with the origin server.

The following example shows how to set Type of Service support to normal:

```
ContentEngine(config)# pre-load set-tos normal
```

**Note** Using the `pre-load dscp` global configuration command takes precedence over any use of the Rules Template configuration commands involving DSCP server configurations.

Table 11-1 describes the DSCP values.

<table>
<thead>
<tr>
<th>DSCP Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0-63&gt;</td>
<td>Valid DSCP value range</td>
</tr>
<tr>
<td>af11</td>
<td>Packets with AF11 dscp (001010)</td>
</tr>
<tr>
<td>af12</td>
<td>Packets with AF12 dscp (001110)</td>
</tr>
<tr>
<td>af13</td>
<td>Packets with AF13 dscp (001110)</td>
</tr>
<tr>
<td>af21</td>
<td>Packets with AF21 dscp (011010)</td>
</tr>
<tr>
<td>af22</td>
<td>Packets with AF22 dscp (010110)</td>
</tr>
<tr>
<td>af23</td>
<td>Packets with AF23 dscp (010110)</td>
</tr>
<tr>
<td>af31</td>
<td>Packets with AF31 dscp (011010)</td>
</tr>
<tr>
<td>af32</td>
<td>Packets with AF32 dscp (011110)</td>
</tr>
<tr>
<td>af33</td>
<td>Packets with AF33 dscp (011110)</td>
</tr>
<tr>
<td>af41</td>
<td>Packets with AF41 dscp (110010)</td>
</tr>
<tr>
<td>af42</td>
<td>Packets with AF42 dscp (110110)</td>
</tr>
<tr>
<td>af43</td>
<td>Packets with AF43 dscp (110110)</td>
</tr>
</tbody>
</table>
Step 13

View the status of the current preloading operation.

The following example shows the status of the current preloading operation after using the `pre-load set-tos` and `pre-load max-bandwidth` commands:

```
ContentEngine# show pre-load
Preloading is enabled
Number of concurrent sessions: 10
Depth level: 4
URL List File: /local1/url.txt
DSCP: set-tos normal
Max Bandwidth: 50000 Kbps
Previous preloading operation will be continued.
Preload will not traverse other domains.
Fetch Domains:
Fetch Suffix:
```
Chapter 11  Configuring Content Preloading and URL Filtering on Standalone Content Engines

Configuring Content Preloading for Standalone Content Engines

**Fetch Directory:**
**No-fetch Domain:**
**No-Fetch Suffix:**
**No-Fetch Directory:**
**Scheduling on all days**
  - **Start Time:** 00:00
  - **End Time:** Till completion

**Step 14** View the statistics associated with the current preloading, after the preload has started.

```
ContentEngine# show statistics pre-load
Statistics of last Preloading operation

Preloading is in progress.
List of preloaded URLs are in /local1/preload_dir/downloaded_urls.

83 objects downloaded, 2842292 bytes transferred.
```

**Step 15** Inform your end users what the URLs of the preloaded files are so that they can use their browsers or media players to access this preloaded content.

---

For an example of how you can verify if the preloaded VOD files are being cached and properly distributed to clients, see the “Verifying That Preloaded VOD Files Are Cached and Properly Distributed to Windows Media Clients” section on page 9-36.

**Proxy Authentication Support for Content Preloading**

Typically preloading works by having a standalone Content Engine preload the content that is specified in the URL list file (the url.txt file). The URL list file also specifies the desired depth level for the URL retrieval (for example, four levels deep). The username and password are available in the URL to perform server authentication if necessary to retrieve a protected object. If there is an intermediate proxy, the preloading of content will not work because the Content Engine is unable to perform proxy authentication.

In releases prior to the ACNS 5.4.1 software release, a typical deployment was that a second Content Engine (CE2) acted as an upstream proxy server for another Content Engine (CE1). CE1 was able to support NTLM authentication if you had specified the `http authentication header` 401 global configuration command on CE2. In the case of NTLM authentication, requests would be authenticated (treating them as WWW-Auth), and if the credentials matched, the CE1 would retrieve the contents and the content would be preloaded on CE1.

In the ACNS 5.4.1 software release, proxy authentication support for content preloading was added. Proxy authentication support for content preloading is supported if the Content Engine has been configured to use any of the following authentication schemes for the outgoing proxy server: NTLM, LDAP, RADIUS, and TACACS+. This feature supports one level of upstream proxy server (that is, the proxy information that is obtained from the preload file only applies to the immediate upstream proxy server).

In the ACNS 5.4.1 software and later releases, if CE1 is configured to use NTLM or basic authentication as the proxy authentication scheme, CE1 is able to perform the initial proxy authentication and then retrieve the requested object from the origin server. The retrieved object can be an unauthenticated or authenticated object.
To support proxy authentication in preloading, content preloading on the Content Engine must be able to accept the proxy user and proxy domain name in order to perform proxy authentication for NTLM or Basic authentication. The following URL format for the preloading process is currently supported:

```
http://user1:user1@10.77.157.131/kerberos/kerberos.htm 1 acns:acns:acns
```

where `user1` is the user name, `user1` is the password, `1` is the depth level, `acns` is the domain name, host name, and domain in which the host resides.

The format for accepting the username, password, and domain name for proxy authentication is as follows:

```
proxy_user:username:proxy_pwd:pwd:proxy_domain_name:domain
```

The username, password, and domain name each can be up to 50 characters in length. This information must be specified in the first line of the preload list file. A colon (:) is used as separator to obtain the information from the preload list file. Consequently, do not use a colon in the password for a user.

If the username, password, and domain name for proxy authentication is not available in the preload file, then the Content Engine will use the URL format to perform the preload operation.

If the Content Engine is configured to use NTLM as the proxy authentication scheme, then the preload file would contain the following type of information:

```
proxy_user: user1:proxy_pwd:user1:proxy_domain_name:acns
```

If the Content Engine is configured to use basic authentication as the proxy authentication scheme, then the preload file would contain the following type of information:

```
proxy_user: tuser1:proxy_pwd:tpass1:proxy_domain_name:null
```

For an example of how to configure proxy authentication for content preloading on standalone Content Engines, see the next section, the “Example of Configuring Proxy Authentication for Content Preloading.”

---

**Example of Configuring Proxy Authentication for Content Preloading**

In the following example, the Content Engine (CE1) is configured to have the Content Engine named CE2 be its outgoing proxy server, and is configured to use NTLM authentication to preload content:

**Step 1**
The Content Engine (CE1) is configured to use NTLM authentication as the proxy authentication method:

```
CE1(config) # ip name-server name-server
CE1(config) # ntlm server host ip-address
CE1(config) # ntlm server enable
```

**Step 2**
Clear the cache and the statistics on CE1:

```
CE1# clear cache
CE1# clear statistics all
```

**Step 3**
Enable the preload feature on CE1:

```
CE1(config) # pre-load enable
```

**Step 4**
Configure the path for the url-list file on CE1:

```
CE1(config) # pre-load url-list-file /local1/preload.txt
```

**Step 5**
Configure the outgoing proxy server (CE2) for CE1:

```
CE1(config) # http proxy outgoing host ip-address port-number
```
Chapter 11 Configuring Content Preloading and URL Filtering on Standalone Content Engines

Step 6 On CE1, enter the following command to preserve the 407 HTTP authentication header:

CE1(config)# http proxy outgoing preserve-407

Step 7 On CE2, enable authentication:

CE2(config)# http proxy incoming port-number

Step 8 On CE1, in the first line of url-list-file, specify the credentials that are to be matched against the NTLM origin server:

CE1# type preload.txt
proxy_user:preload:proxy_pwd:preload1:proxy_domain_name:acns
http://www.yahoo.com/
http://10.77.157.60/

In the above preload file, the username is preload, the password is preload1, and the domain name is acns. These credentials will be sent to the NTLM origin server when CE1 receives a 407 message from the outgoing proxy server (CE2).

Step 9 Force a preload operation on CE1:

CE1# preload force

Step 10 After the forced preload operation has been completed on CE1, verify that all of the objects have been preloaded properly on CE1.

a. On CE1, check the latest_preloaded_objects files in the /local/local1/preload_dir directory.

b. On CE1, check the latest_preload_error file to verify that there are no entries indicating an error because of a proxy authentication failure or a 407 message.

c. On CE1, enter the following command to verify that the objects have been properly preloaded on CE1:

CE1# show statistics preload
Preloading was initiated by force.
Preloading started at Thu Mar 13 06:42:40 2003
Preloading ended at Thu Mar 13 06:42:53 2003
List of preloaded URLs are in /local1/preload_dir/latest_preloaded_objects.
Preload errlog is /local1/preload_dir/latest_preload_error.

Number of invalid entries in URL list file = 0
Total number of preloaded objects = 1
Total number of preloaded bytes = 570

d. On CE2, enter the following command to verify that the user (the user with the name preload) is listed in the authentication cache on CE2:

CE2# show http-authcache

Step 11 From a web client that has CE1 as its proxy server, use the browser to issue a request for http://10.77.157.60.

Step 12 Verify that this client request is served from the cache on CE1:

a. On CE1, enter the following command:

CE1# show statistics http savings

b. Check the command output to verify that the hit counter has been incremented on CE1.
c. On CE1, enter the following command to verify that the objects have been properly preloaded on CE1:

```
CE1# show statistics preload
Preloading was initiated by force.
Preloading started at Thu Mar 13 06:42:40 2003
Preloading ended at Thu Mar 13 06:42:53 2003
List of preloaded URLs are in /local1/preload_dir/latest_preloaded_objects.
Preload errlog is /local1/preload_dir/latest_preload_error.
```

Number of invalid entries in URL list file = 0
Total number of preloaded objects = 1
Total number of preloaded bytes = 570

d. On CE2, enter the following command to verify that the user (the user with the name preload) is listed in the authentication cache on CE2:

```
CE2# show http-authcache
```

Step 13
From a web client that has CE1 as its proxy, use the browser to issue a request for http://10.77.157.60.

Step 14
Verify that this client request is served from the cache on CE1:

a. On CE1, enter the following command:

```
CE1# show statistics http savings
```

b. Check the command output to verify that the hit counter has been incremented on CE1.

---

**Stopping or Resuming Content Preloading on Standalone Content Engines**

To stop a preload process that is currently in progress on a standalone Content Engine, use the `no pre-load enable` global configuration command.

If the content preloading is not completed before the scheduled end time, you can resume the preloading process to capture content using the `pre-load resume` global configuration command. Using this command allows you to resume downloading from the breakpoint of the previous preload, instead of starting again from the very beginning of the preload URL list file.

**Note**

If the `pre-load resume` global configuration command is not set up on the Content Engine and content preloading is aborted before the scheduled end time, the next scheduled content preloading starts from the beginning of the preload URL list file.

---

**Configuring URL Filtering on Standalone Content Engines**

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 Content Engines, organizations realize an immediate return on investment as a result of increased productivity and recaptured network bandwidth, while reducing legal liability.

Table 11-3 lists the various URL filtering schemes that you can configure a standalone Content Engine to use in order to control client access to websites.
Chapter 11  Configuring Content Preloading and URL Filtering on Standalone Content Engines

Configuring URL Filtering on Standalone Content Engines

See Table B-6 for a list of the URL filtering schemes (for example, SmartFilter or Websense) that are supported for different protocols.

Although only one form of URL filtering scheme can be active at a time per protocol, many URL filtering schemes can be supported at one time. For example, if an N2H2 filter is applied to HTTP requests, then no other URL filtering scheme (for instance, Websense or SmartFilter) can be applied to this protocol. However, you could apply the local list URL filtering scheme (good lists and bad lists) to the streaming media protocols (WMT client requests and client requests over RTSP). The scheme enabled for a particular protocol is independent from that of other protocols.

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

To ensure that URL filtering applies to every URL that passes through the Content Engine, disable all bypass features. By default, load bypass is enabled.

- To use the Content Engine GUI to disable load bypass manually, choose **Caching > Bypass** and then click the **Load Bypass Off** radio button in the Bypass window.
- To disable load bypass manually through the Content Engine CLI, use the **bypass load** global configuration command.

```bash
ContentEngine(config)# no bypass load enable
```

- To disable error handling manually through the Content Engine CLI, use the **error-handling send-cache-error** or **error-handling reset-connection** global configuration command. (By default, error handling is enabled on the Content Engine.)

```bash
ContentEngine(config)# no error-handling send-cache-error
ContentEngine(config)# no error-handling reset-connection
```

When both RADIUS authentication and URL filtering are enabled on the Content Engine, the user Filter-Id attribute in the RADIUS server database can be configured to bypass URL filtering.

The following example shows an example of a user Filter-Id attribute entry in the RADIUS server database:

```bash
test
    Password = "test"
    Service-Type = Framed-User,
    Filter-Id = "No-Web-Blocking"
```
The Filter-Id attribute is defined as either No-Web-Blocking or Yes-Web-Blocking. Yes-Web-Blocking means that the request is subject to URL filtering, and No-Web-Blocking means that the request is not subject to URL filtering. If blocking is not specified, Yes-Web-Blocking is the default RADIUS filter.

**Note**

For more information about the use of a RADIUS server for authentication purposes and URL filtering, see the “Understanding RADIUS Authentication and Authorization” section on page 17-6.

---

### Configuring Local List URL Filtering on Standalone Content Engines

You can configure standalone Content Engines to deny client requests for URLs that are listed in a badurl.lst file, or configure them to fulfill only requests for URLs in a goodurl.lst file. The use of local list files (URL lists) applies to HTTP (HTTP, HTTPS-over-HTTP, and FTP-over-HTTP) as well as RTSP streaming media protocol. This type of URL filtering is referred to as local list URL filtering.

**Tip**

Only one good sites file or one bad sites file can be active at one time per protocol.

The local list file for each protocol should not contain URLs that belong to other protocols. For instance, the HTTP local list file should contain only the following types of URLs: HTTP, HTTPS, or FTP URLs. In the ACNS 5.3.1 software and later releases, the WMT local list file can contain RTSP URLs.

**Caution**

If the size of the local list file becomes too large, it can adversely effect proxy performance, because the local list file is loaded into memory when local list filtering is enabled. If the file size is larger than 5 MB, a warning message appears, but the ACNS software does not enforce size limits for the local list file. It is your responsibility to track the local list file size and ensure that it does not become so large that it degrades performance.

You can configure a standalone Content Engine to use local list URL filtering to filter the following types of client requests for content:

- Requests over HTTP (HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests)
- RealMedia requests (IETF standard RTSP protocol with RealNetworks proprietary extensions)
- WMT requests (MMS-over-HTTP and RTSP-over-RTP for Windows Media 9 clients and Windows Media 9 servers)

Filtering for native FTP and native HTTPS requests is not supported.

Support for RTSP-over-RTP (as referred to as *WMT RTSP requests*) for Windows Media 9 players is available in the ACNS 5.3.1 software and later releases. For WMT RTSP requests, there are three possible protocol prefixes: rtsp, rtspu, and rtspt.

If the user enters rtsp: as the protocol prefix for the URL, the Windows Media 9 player can choose to use RTSP or RTSPU. If the rtsp bad file has a URL of rtsp://hostname/pathname and the user’s URL request is rtspt://hostname/pathname, then the RTSP request from a Windows Media 9 player might get through the URL filtering. Special URL filtering for RTSP requests from Windows Media 9 players is available in the ACNS 5.3.1 software and later releases.

For WMT URL filtering, filtering for RTSP URLs (rtp://) is only supported; there is no separate filtering support for RTSPT and RTSPU URLs. However, if you configure an RTSP URL in the badurl.lst file, then it will block both the RTSPT and RTSPU URLs.
To use the Content Engine CLI to configure local list URL filtering on a standalone Content Engine, use the `url-filter` global configuration commands. In the ACNS 5.3.1 software release, the `url-filter` command was modified to support local list URL filtering for RTSP requests from Windows Media 9 players. In the ACNS 5.2.x software and earlier releases, the `url-filter` command options were as follows:

```
ContentEngine(config)# url-filter ?
    http For requests over HTTP
    rtsp For requests over RTSP
    wmt  For WMT requests
```

In the ACNS 5.3.1 software and later releases, the `url-filter` command options are as follows:

```
ContentEngine(config)# url-filter ?
    http For requests over HTTP and MMS over HTTP
    rtsp For requests over RTSP - applies to real proxy, real server and cisco streaming engine
    wmt  For WMT requests - applies to MMS and RTSP
```

Local list URL filtering is the only supported filtering method for WMT requests (MMS-over-HTTP and RTSP requests from WMT clients) and RTSP requests (requests from RealMedia players). The third-party URL filtering methods (N2H2, SmartFilter, and Websense software) are not supported for WMT and RTSP requests. For HTTP requests, the local list URL filtering method as well as N2H2, SmartFilter, and Websense URL filtering is supported. See Table B-4 for a list of the kinds of URL filtering that are supported for the different protocols.

Table 11-4 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for HTTP requests (HTTP, FTP-over-HTTP, MMS-over-HTTP, and HTTPS-over-HTTP requests).

<table>
<thead>
<tr>
<th>CLI Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url-filter http bad-sites-deny enable</td>
<td>Configures the Content Engine to deny client requests to URLs in the HTTP bad site list.</td>
</tr>
<tr>
<td>url-filter http bad-sites-deny file filename</td>
<td>Specifies the filename of the HTTP bad site list.</td>
</tr>
<tr>
<td>url-filter http good-sites-allow enable</td>
<td>Configures the Content Engine to permit client requests to URLs in the HTTP good site list.</td>
</tr>
<tr>
<td>url-filter http good-sites-allow file filename</td>
<td>Specifies the filename of the HTTP good site list.</td>
</tr>
</tbody>
</table>
Table 11-5 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for requests over RTSP. This type of URL filtering is used with RealProxy, which is a backend RTSP server that is running on the standalone Content Engine. With registered Content Engines, RealProxy, RealSubscriber, and the Cisco Streaming Engine that are running on the registered Content Engine use this type of URL filtering.

Table 11-5 Configuring Standalone Content Engines to Use Local List URL Filtering for Requests over RTSP

<table>
<thead>
<tr>
<th>CLI Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url-filter rtsp bad-sites-deny enable</td>
<td>Configures the Content Engine to deny client requests to URLs in the RTSP bad site list.</td>
</tr>
<tr>
<td>url-filter rtsp bad-sites-deny file filename</td>
<td>Specifies the filename of the RTSP bad site list.</td>
</tr>
<tr>
<td>url-filter rtsp good-sites-allow enable</td>
<td>Configures the Content Engine to permit client requests to URLs in the RTSP good site list.</td>
</tr>
<tr>
<td>url-filter rtsp good-sites-allow file filename</td>
<td>Specifies the filename of the RTSP good site list.</td>
</tr>
</tbody>
</table>

Table 11-6 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for WMT requests (WMT requests over RTSP). If you configure an RTSP URL in a WMT bad site list, then it blocks both the RTSPT and RTSPU URLs as well as the RTSP URL that is specified in the bad site list.

Table 11-6 Configuring Standalone Content Engines to Use Local List URL Filtering for WMT Requests

<table>
<thead>
<tr>
<th>CLI Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url-filter wmt bad-sites-deny enable</td>
<td>Configures the Content Engine to deny client requests to URLs in the WMT bad site list.</td>
</tr>
<tr>
<td>url-filter wmt bad-sites-deny file filename</td>
<td>Specifies the filename of the WMT bad site list.</td>
</tr>
<tr>
<td>url-filter wmt good-sites-allow enable</td>
<td>Configures the Content Engine to permit client requests to URLs in the WMT good site list.</td>
</tr>
<tr>
<td>url-filter wmt good-sites-allow file filename</td>
<td>Specifies the filename of the WMT good site list.</td>
</tr>
</tbody>
</table>

In the ACNS 5.3.1 software and later releases, the `url-filter wmt` global configuration commands apply to RTSP. URL filtering for RTSP requests is used when the client is a Windows Media 9 player and the server is a Windows Media 9 server. If an earlier version of the Windows Media player is used (for example, Windows Media 7 players), the MMS-over-HTTP protocol is used instead of the RTSP protocol to service the content request from the Windows Media player.
Example of Configuring URL Filtering with Local URL Lists

To configure a standalone Content Engine to use a local list file to deny client requests for specific HTTP URLs, follow these steps:

**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 be delimited with carriage returns.

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

**Tip** We recommend creating a separate directory under local1 to hold the bad lists, for example, /local1/filtered_urls.

**Step 3** Point the Content Engine to the bad URL list.
ContentEngine(config)# url-filter http bad-sites-deny file local/local1/badurl.lst

**Step 4** Configure the Content Engine to actively deny the URLs.
ContentEngine(config)# url-filter http bad-sites-deny enable

**Step 5** Reload the new bad site list on the standalone Content Engine.
ContentEngine# url-filter local-list-reload http

To configure a standalone Content Engine to use a local list file permit specific HTTP URLs to the exclusion of all other URLs, follow these steps:

**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 be delimited with carriage returns.

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

**Tip** We recommend creating a separate directory under local1 to hold the good lists, for example, /local1/filtered_urls.

**Step 3** Point the Content Engine to the goodurl.lst file.
ContentEngine(config)# url-filter http good-sites-allow file local/local1/goodurl.lst

**Step 4** Configure the Content Engine to actively permit only the good URLs.
ContentEngine(config)# url-filter http good-sites-allow enable

**Step 5** Reload the new good site list on the standalone Content Engine.
ContentEngine# url-filter local-list-reload http
**Reloading Local List Files on Standalone Content Engines**

When you update the badurl.lst or goodurl.lst file, use the `url-filter local-list-reload` EXEC command to reload the good site or bad site lists on the standalone Content Engine if the URL list feature is enabled.

```
url-filter local-list-reload {http | rtsp | wmt}
```

The syntax is as follows:

- `http` reloads the new local lists for HTTP requests (HTTP, FTP-over-HTTP, MMS-over-HTTP, and HTTPS-over-HTTP requests).
- `rtsp` reloads the local lists for requests over RTSP (requests from RealMedia clients).
- `wmt` reloads the local lists for WMT requests (RTSP-over-RTP [the standard IETF RTSP protocol with Microsoft proprietary extensions] requests from Windows Media 9 players).

The following examples shows how to reload new good site or bad site lists on a standalone Content Engine:

```
ContentEngine# url-filter local-list-reload http
ContentEngine# url-filter local-list-reload rtsp
ContentEngine# url-filter local-list-reload wmt
```

**Creating Custom Blocking Messages**

In the case of local list URL filtering, you can configure standalone Content Engines to return a customized blocking message to the client that requested content that is served through the Content Engine. 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. The following is an example of the contents of the block.html file:

```
<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>
</BLINK>
<FONT COLOR="#0080FF">Blocked Page</FONT>
</I></B></CENTER>
</H1>
<p>
<IMG src="/content/engine/blocking/url/my.gif">
<p>
This page is blocked by the Content Engine.
```

If the block.html file is updated, it will automatically display its new message without requiring you to reenter the `url-filter http custom-message` command.
In the following example, a block.html file displays this custom message when the standalone Content Engine intercepts a request to the blocked site:

This page is blocked by the Content Engine

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 preceding example.

To enable the customized blocking message, use the `url-filter http custom-message` global configuration command and specify the directory name. To disable the custom message, use the `no url-filter http custom-message` command.

You can enable and disable the `url-filter http custom-message` command 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.

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

### Configuring Standalone Content Engines for N2H2 URL Filtering

N2H2 is a globally deployed URL-filtering solution that can filter HTTP, FTP, or HTTPS requests based on destination hostname, destination IP address, and username and password. N2H2 relies on a sophisticated URL database exceeding 15 million sites and is organized into over 40 categories using both Internet technology and human review. See [http://www.n2h2.com](http://www.n2h2.com) for further information on N2H2 filtering products.

**Note**

See Table B-4 for a list of the kinds of URL filtering that are supported for the different protocols with an N2H2 server.

N2H2 supports three filtering methods. Table 11-7 lists the N2H2 features supported by the Content Engine. One N2H2 server can support multiple Content Engines simultaneously.

<table>
<thead>
<tr>
<th>N2H2 Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global filtering</td>
<td>Applies filtering to all HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over HTTP requests.)</td>
</tr>
<tr>
<td>User-based filtering</td>
<td>Applies filtering to specific users or groups.</td>
</tr>
<tr>
<td>Client IP-based filtering</td>
<td>Applies filtering to specific client IP addresses.</td>
</tr>
<tr>
<td>Transparent authentication</td>
<td>Performs transparent authentication by passing back the initial response header to the client using the HTML page in IFP responses.</td>
</tr>
</tbody>
</table>
Standalone Content Engines can use an N2H2 enterprise server as a filtering engine and enforce the filtering policy configured on the N2H2 server. (See Figure 11-1.) The standalone Content Engine and the N2H2 server use Internet Filtering Protocol (IFP) Version 2 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 page where a blocking message is displayed or proceeds with normal HTTP processing by sending the URL request to an origin server.

**Figure 11-1  N2H2 Filtering**

![Diagram of N2H2 Filtering](image)

**Note**
URL filtering using an N2H2 server is applied to HTTP traffic (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) before the Rules Template is applied, regardless of whether the requested object is in the cache or not. See Table B-4 for the kinds of URL filtering that are supported for the different protocols with N2H2.

To configure a standalone Content Engine to use an external N2H2 server for URL filtering, follow these steps:

**Step 1** Display the URL filtering schemes that are currently enabled on this Content Engine for requests over HTTP.

```
ContentEngine# show url-filter http
```

**Step 2** Make sure that no other URL filtering scheme (for example, Websense or SmartFilter software) is currently enabled for requests over HTTP. Only one URL filtering scheme per protocol can be active at a time.
Step 3 Configure the Content Engine to use an external N2H2 server for URL filtering by using the `url-filter http N2H2 server` global configuration command.

   a. Specify the necessary information about the external N2H2 server (for example, its IP address).

      `url-filter http N2H2 server {[hostname | ip-address]} [port portnum [timeout seconds]]`

      • `hostname` is the hostname of the external N2H2 server.
      • `IP address` is the IP address of the external N2H2 server.
      • `portnum` is the port number (1–65535) to which the Content Engine sends the IFP requests to the specified N2H2 server. The default port number is 4005.
      • `seconds` is the number of seconds (1–120) that the Content Engine is to wait for an IFP response from the N2H2 server before timing out the connection. The default timeout is 5 seconds.

In the following example, the Content Engine is configured to use an N2H2 server that has an IP address of 172.16.22.10. The Content Engine will send IFP requests to this N2H2 server on port 4008 and will wait for up to 100 seconds for an IFP response from this server before timing out the connection:

```
ContentEngine(config)# url-filter http N2H2 server 172.16.22.10 port 4008 timeout 100
```

The server IP address and port number configured on the standalone Content Engine must match the IP address of the N2H2 server and the port that the 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 requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) and either block or allow all HTTP traffic based on the `allowmode` option configuration.

**Note** The `url-filter http N2H2 server` global configuration 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.

Step 4 Enable the N2H2 URL filtering scheme on this Content Engine.

```
ContentEngine(config)# url-filter http N2H2 enable
```

Step 5 Allow HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) to pass through when the N2H2 server is enabled but the Content Engine has problems communicating with the N2H2 server by using the `url-filter http N2H2 allowmode enable` global command.

- When `allowmode` is enabled, the Content Engine allows all HTTP traffic to continue through it (it proceeds with normal traffic processing) even if it fails to receive responses from the N2H2 server.
- When `allowmode` is disabled, the Content Engine blocks all HTTP traffic that is served through it if it fails to receive responses from the N2H2 server.

By default, `allowmode` is enabled. You can configure the `allowmode` option with or without N2H2 being enabled; it is independent of the N2H2 server configuration. The Content Engine adopts the new configuration for `allowmode` if N2H2 URL filtering is already being used.

Step 6 Display the request-reply statistics for the communication between the Content Engine and the N2H2 server.

```
ContentEngine# show statistics url-filter http N2H2
```
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. The statistics shown can be cleared using the `clear statistics url-filter http N2H2` and `clear statistics all` EXEC commands. The `clear statistics url-filter http N2H2` EXEC command resets the statistics counters for the N2H2 server. All the statistics counters are reset to 0.

**Note**
Go to [http://www.n2h2.com](http://www.n2h2.com) for further information on N2H2 filtering configuration and policies.

---

**Configuring Standalone Content Engines for Websense URL Filtering**

Standalone Content Engines can use a remote Websense enterprise server as a filtering engine and enforce the filtering policy configured on the Websense server. As Figure 11-2 shows, the remote Websense server runs on a separate system (Host A) from the local Websense server and communicates with the standalone Content Engine over the network.

![Figure 11-2 URL Filtering with Websense Servers](image)

You can also configure a standalone Content Engine to use the integrated Websense server. The integrated Websense server is an internal server that runs on the Content Engine, and is referred to as the local Websense server.

**Note**
In the ACNS 5.1 software and earlier releases, only one Websense server is supported. In the ACNS 5.2.1 software and later releases, up to two Websense servers are supported. For more information on this topic, see the next section, “About Websense Server Failover.”

The ACNS 5.4.1 software and later releases support the Websense 5.5.2 software. The ACNS 5.3.x software supports the Websense 5.2.0 software.
Chapter 11 Configuring Content Preloading and URL Filtering on Standalone Content Engines

About Websense Server Failover

In the ACNS 5.2.1 software and later releases, the Websense server failover feature is supported. This feature allows you to configure a Content Engine to use up to two Websense servers for failover purposes (one primary and one secondary server) for URL filtering. Table 11-8 lists the supported Websense server failover configurations.

<table>
<thead>
<tr>
<th>Supported Configurations</th>
<th>Local (Internal) Websense Server</th>
<th>Remote Websense Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option A</td>
<td>The local Websense server is disabled on the Content Engine.</td>
<td>The primary Websense server is running on an external host (for example, Host A). The secondary Websense server is running on a second external host (for example, Host B).</td>
</tr>
<tr>
<td>Option B</td>
<td>The local Websense server is acting as the primary Websense server.</td>
<td>The secondary Websense server is running on an external host.</td>
</tr>
<tr>
<td>Option C</td>
<td>The local Websense server is acting as the secondary Websense server.</td>
<td>The primary Websense server is running on an external host.</td>
</tr>
</tbody>
</table>

The order in which you configure the Websense servers determines which server is designated the primary Websense server. The first configured Websense server is designated the primary server. Configuration of a secondary Websense server is optional. For an example of how to configure Websense server failover for a standalone Content Engine, see the “Example of Configuring Websense Server Failover and URL Filtering” section on page 11-30.

About Websense Services

The ACNS 5.4.x software and later releases supports the Websense 5.5.2 software. The following services are supported in the Websense 5.5.2 software:

- Policy Server
- Employee Internet management (EIM) Server
- Local Network Agent
- Local RADIUS Agent
- Local eDirectory Agent
- Local Logon Agent (support added in the ACNS 5.4.1 software release)
- Local User Service

Note

When you install or activate additional Websense components for the integrated (local) Websense server, the ACNS software requires a minimum of 1 GB of RAM.

Note

The term local Websense server is still used to refer collectively to these Websense processes that are running internally on the Content Engine. These Websense processes are also called services.
In the Websense GUI Manager Version 5.5 and 6.1 that is supported by the Websense 5.5.2 software, you can configure the RADIUS and eDirectory agent through the GUI as well as through CLI commands. Consequently, in the ACNS 5.4.1 software release, all of the global configuration CLI commands that were related to configuring the RADIUS and eDirectory agent have been removed from the ACNS CLI command set. However, the CLI commands that are used to active the RADIUS and eDirectory agents have been retained (the `websense-server service radius-agent activate` and the `websense-server service edirectory-agent activate` global configuration commands).

- The following global configuration CLI commands for configuring the RADIUS agent were removed in the ACNS 5.4.1 software release:
  - `websense-server service radius-agent incoming [auth-number] [acct-port port number]`
  - `websense-server service radius-agent outgoing [host remote-RADIUS-server IP-address] [auth-port port number] [acct-port port number]`

- The following global configuration CLI commands for configuring the eDirectory agent were removed in the ACNS 5.4.1 software release:
  - `websense-server service edir-agent edir-server [administrative-dn administrative-distinguished-name] [host remote-eDirectory-server IP-address] [root-context root-context]`

If the Content Engine is running the ACNS 5.4.1 software or a later release, and if you attempt to configure the RADIUS agent or the eDirectory agent through the CLI by entering any of the preceding global configuration commands, the command is nullified. You will not receive an error message if you enter one of these nullified commands.

The global configuration command for configuring the eDirectory administrative password was retained in the ACNS 5.4.1 software release:

```
ContentEngine(config)# websense-server service edirectory-agent edir-server administrative-passwd password
```

With the Websense 5.5.2 software, you can use a local or remote Websense Policy Server to activate the local EIM Server, the local RADIUS Agent, the local eDirectory Agent, the local Network Agent, the local Logon Agent, and the local User Service individually on a Content Engine. (See Table 11-9.)
### Table 11-9  Services of the Local Websense 5.5.2 Server Supported in the ACNS 5.4.1 Software or Later

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Server</td>
<td>Hosts all of the policy information that you have configured through the external Websense Manager GUI. Communicates the policy information to the other services of the local Websense server (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service). The local (internal) Policy Server or the specified remote Policy Server must be running, before you can activate the local EIM Server, the local Network Agent, the local Logon Agent, or the local User Service on the Content Engine.</td>
</tr>
<tr>
<td>Local EIM Server</td>
<td>Provides the URL filtering functionality when used with proxy servers, firewalls, and caching appliances.</td>
</tr>
</tbody>
</table>
| Local Network Agent       | Enables URL filtering of requests that use protocols other than HTTP, HTTPS-over-HTTP, and FTP-over HTTP. If the local Network Agent is activated on the Content Engine, the Network Agent can filter incoming requests from the following protocols and applications:  
  - Database applications such as SQL Net  
  - File transfer applications such as FTP and Gopher  
  - Instant messaging and chat applications such as Yahoo Messenger, and MSN Messenger  
  - Mail and collaborative tools such as POP3, SMTP, and NetMeeting  
  - Network operating system applications such as Daytime, finger, NTP, SSH, and Telnet  
  - Remote access applications such as VNC and pcANYWHERE  
  - Streaming media applications such as RTSP, Windows Media, and Liquid Audio  
  - Other (for example, Network News Transfer Protocol [NNTP])                                                                                                                                                                                                                                                                                                     |
| Local RADIUS Agent        | Enables URL filtering based on user-based or group-based policies for users who are authenticated through an external RADIUS server. This agent transparently identifies the users who access the network and are authenticated through the RADIUS authentication scheme. When the Content Engine is supplied with this information, the Content Engine can apply policies to users and groups of the users who access the network remotely.  
  This agent acts as a proxy that forwards the RADIUS messages between the RADIUS client and the external RADIUS server. For the local RADIUS Agent to work properly, you must have configured the RADIUS settings (for example, the IP address of the external RADIUS server) on the Content Engine, as described in the “Specifying RADIUS Authentication Settings for Standalone Content Engines” section on page 17-10.                                                                                                 |
Configuring URL Filtering on Standalone Content Engines

Local eDirectory Agent
Enables URL filtering based on user-based or group-based policies for users who are authenticated through LDAP. This agent works in conjunction with the Novell eDirectory to transparently identify users who access the network and are authenticated through the LDAP authentication scheme. When the Content Engine is supplied with this information, the Websense filtering service can filter requests based on the policies applied to the users or groups.

This agent uses LDAP to gather the user login session information from the Novell eDirectory, which authenticates users logging in to the network. This agent associates each authenticated user with the IP address. With the help of the Websense local User Service, the local eDirectory Agent supplies this information to the Websense filtering service. For this local eDirectory Agent to work properly, you must have configured settings such as the administrative distinguished name, using the Websense Manager GUI (Version 5.5, or 6.1 or later).

Local User Service
Enables URL filtering based on user-based or group-based policies. If you are using a user service and you want to configure user-based or group-based URL filtering using a Windows NT directory, then you must use the external user service on a Windows machine.

Local Logon Agent
Enables URL filtering based on user-based or group-based policies for users as they log on to the network through a Windows client machine. This agent’s associated logon application captures logon sessions as users log on to Windows domains in a network. The Logon Agent communicates with the Websense User Service to provide up-to-date user logon session information to Websense for filtering purposes.

The Logon Agent identifies users in a real-time manner, as they log on to a domain, whereas a Domain Controller (DC) Agent identifies users by periodically querying domain controllers and workstations. Because the Logon Agent identifies users in a real-time manner, the Websense Filtering Service can accurately filter Internet access based on the policies assigned to particular users, groups, workstations, or networks.

The Logon Agent user identification process is the identification process that the Logon Agent relies on for the self-identification of the Windows client machine. Typically, a logon script in a shared network location invokes a process on the client machines called LogonApp.exe. LogonApp.exe has two operation modes: persistent mode and nonpersistent mode.

- **Persistent mode**: LogonApp.exe runs as a background task on the client machine, and periodically sends the username and password pairs to the Logon Agent. The interval is determined by the Query Interval (persistent mode) setting on the Websense Enterprise Manager. If a logout script has been configured to register when a user logs out, then the LogonApp.exe sends the logout information to the Logon Agent at that time.

- **Nonpersistent mode**: LogonApp.exe contacts the Logon Agent once when a user logs on. The user logon session is stored in the Logon Agent’s local memory. The Logon Agent’s user map is subject to the Entry Lifetime (non-persistent mode) setting on the Websense Enterprise Manager.

The Logon Agent obtains logon session information from its associated logon application, LogonApp.exe, and stores the username and IP address pairs in a user map in local memory and in the AuthServer.journal file. IP addresses, rather than usernames, are the key element in tracking logon sessions because it is possible for the same user to log on to a network domain from multiple workstations. Each time the size of the AuthServer.journal file reaches 1 MB, the Logon Agent backs up the contents to the AuthServer.bak file on its hard disk.

The Logon Agent, like the eDirectory agent, transparently identifies users. Support for the Logon Agent was added in the ACNS 5.4.1 software release.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local eDirectory Agent</td>
<td>Enables URL filtering based on user-based or group-based policies for users who are authenticated through LDAP. This agent works in conjunction with the Novell eDirectory to transparently identify users who access the network and are authenticated through the LDAP authentication scheme. When the Content Engine is supplied with this information, the Websense filtering service can filter requests based on the policies applied to the users or groups. This agent uses LDAP to gather the user login session information from the Novell eDirectory, which authenticates users logging in to the network. This agent associates each authenticated user with the IP address. With the help of the Websense local User Service, the local eDirectory Agent supplies this information to the Websense filtering service. For this local eDirectory Agent to work properly, you must have configured settings such as the administrative distinguished name, using the Websense Manager GUI (Version 5.5, or 6.1 or later).</td>
</tr>
<tr>
<td>Local User Service</td>
<td>Enables URL filtering based on user-based or group-based policies. If you are using a user service and you want to configure user-based or group-based URL filtering using a Windows NT directory, then you must use the external user service on a Windows machine.</td>
</tr>
<tr>
<td>Local Logon Agent</td>
<td>Enables URL filtering based on user-based or group-based policies for users as they log on to the network through a Windows client machine. This agent’s associated logon application captures logon sessions as users log on to Windows domains in a network. The Logon Agent communicates with the Websense User Service to provide up-to-date user logon session information to Websense for filtering purposes. The Logon Agent identifies users in a real-time manner, as they log on to a domain, whereas a Domain Controller (DC) Agent identifies users by periodically querying domain controllers and workstations. Because the Logon Agent identifies users in a real-time manner, the Websense Filtering Service can accurately filter Internet access based on the policies assigned to particular users, groups, workstations, or networks. The Logon Agent user identification process is the identification process that the Logon Agent relies on for the self-identification of the Windows client machine. Typically, a logon script in a shared network location invokes a process on the client machines called LogonApp.exe. LogonApp.exe has two operation modes: persistent mode and nonpersistent mode. <strong>Persistent mode</strong>: LogonApp.exe runs as a background task on the client machine, and periodically sends the username and password pairs to the Logon Agent. The interval is determined by the Query Interval (persistent mode) setting on the Websense Enterprise Manager. If a logout script has been configured to register when a user logs out, then the LogonApp.exe sends the logout information to the Logon Agent at that time. <strong>Nonpersistent mode</strong>: LogonApp.exe contacts the Logon Agent once when a user logs on. The user logon session is stored in the Logon Agent’s local memory. The Logon Agent’s user map is subject to the Entry Lifetime (non-persistent mode) setting on the Websense Enterprise Manager. The Logon Agent obtains logon session information from its associated logon application, LogonApp.exe, and stores the username and IP address pairs in a user map in local memory and in the AuthServer.journal file. IP addresses, rather than usernames, are the key element in tracking logon sessions because it is possible for the same user to log on to a network domain from multiple workstations. Each time the size of the AuthServer.journal file reaches 1 MB, the Logon Agent backs up the contents to the AuthServer.bak file on its hard disk. The Logon Agent, like the eDirectory agent, transparently identifies users. Support for the Logon Agent was added in the ACNS 5.4.1 software release.</td>
</tr>
</tbody>
</table>
The Websense 5.5.2 software, which is supported in the ACNS 5.4.1 software and later releases, includes an additional agent called the logon agent.

Table 11-10 lists the CLI commands that are related to configuring the Websense 5.5.2 software on standalone Content Engines.

### Table 11-10 Websense Server-Related CLI Commands

<table>
<thead>
<tr>
<th>CLI Command Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>websense-server service policy local activate</td>
<td>Activates the local Policy Server on the Content Engine.</td>
</tr>
<tr>
<td>websense-server service policy remote [host remote-policy-server ip-address] [port remote-policy-server port-number]]</td>
<td>Specifies the remote Policy Server to be used to activate the local EIM Server, the local Network Agent, and the local User Service on the Content Engine. The default port number is 55806.</td>
</tr>
<tr>
<td>websense-server service eim activate</td>
<td>Activates the local EIM Server on the Content Engine. Use the no form of this command to deactivate it.</td>
</tr>
<tr>
<td>websense-server service network-agent activate</td>
<td>Activates the local Network Agent on the Content Engine. Use the no form of this command to deactivate it.</td>
</tr>
<tr>
<td>websense-server service user activate</td>
<td>Activates the local User Service on the Content Engine. Use the no form of this command to deactivate it.</td>
</tr>
<tr>
<td>websense-server service radius-agent activate</td>
<td>Activates the local RADIUS Agent on the Content Engine. Use the no form of this command to deactivate it.</td>
</tr>
<tr>
<td>websense-server service edir-agent activate</td>
<td>Activates the local eDirectory Agent on the Content Engine. Use the no form of this command to deactivate it.</td>
</tr>
<tr>
<td>websense-server service edir-agent edir-server administrative-passwd password</td>
<td>Specifies the administrative password that the Content Engine uses to contact the external eDirectory Server to request a database search. The following example shows how to use the administrative-passwd command option to specify default244 as the administrative password: ContentEngine(config)# websense-server service edir-agent edir-server administrative-passwd default244</td>
</tr>
<tr>
<td>websense-server service logon-agent activate</td>
<td>Activates the local Logon Agent on the Content Engine. Use the no form of this command to deactivate it. This command was added in the ACNS 5.4.1 software release.</td>
</tr>
</tbody>
</table>

**Note** In the ACNS 5.2 software, the websense-server ip-address and websense-server user-server external global configuration commands are deprecated.
In the ACNS 5.2.1 software and later releases, you can configure a Content Engine to use up to two Websense servers for URL filtering.

To configure a Content Engine for Websense URL filtering, determine the type of Websense server configuration that you want to use for URL filtering:

- To use two Websense servers (the local Websense server and an external Websense server, or two external Websense servers), see the “Example of Configuring Websense Server Failover and URL Filtering” section on page 11-30.
- To use only the local Websense server, see the “Configuring URL Filtering with a Local Websense Server” section on page 11-33.
- To use only an external Websense server, see the “Configuring Websense URL Filtering with External Websense Servers” section on page 11-36.

Only one URL filtering scheme per protocol can be active at a time. In order to enable Websense URL filtering for requests over HTTP, you should make sure that no other URL filtering scheme is configured per protocol. To display the URL filtering schemes that are currently enabled on this Content Engine for HTTP requests (HTTP, FTP-over-HTTP, and HTTPS-over-HTTP), use the `show url-filter http` EXEC command. See Table B-4 for the kinds of URL filtering that are supported for different protocols with a Websense server.

**Note**

In the ACNS 5.4.x software and later releases, Websense server Version 5.5.2 is supported on all Cisco Content Engine platforms. For more detailed information about configuring the Websense software, go to the following website: [http://www.websense.com](http://www.websense.com).

Websense software provides an image of the Websense server that resides in the `/local1/WebsenseEnterprise/EIM` directory on the Content Engine. All the executables as well as the configuration and logging files are stored in this directory.

In the ACNS 5.4.1 software and later releases, Websense installation output is logged to a file named `ws_history.log`. This particular log file resides in the `/local1/logs/urlfilter/websense` directory on the Content Engine.

In the ANCS 5.4.1 software and later releases, you can enter the `clear websense` EXEC command to remove the existing Websense configuration. This command can be useful if you want to remove a corrupted Websense configuration from a standalone Content Engine.

When the Websense server is enabled and the Websense URL database is downloaded to the Content Engine for the first time, CPU usage will be high. Therefore, it is recommended that you enable the Websense server during off-peak times or at times of low network traffic; otherwise, other processes running on the Content Engine may be affected. If one of the Websense processes exits, the local Websense server is automatically restarted on the Content Engine.

To download the Websense components, such as Explorer, Manager, and Reporter, or to obtain an evaluation key for use with the local Websense server that runs on the standalone Content Engine, access the following URL and follow the sequence of steps:

Configuring Ports for the Websense Server

The Websense process requires that four ports be open for connections either from processes internal to the Content Engine or from external processes such as the Websense Manager. (See Table 11-11.)

Table 11-11   Configuration of Ports for the Websense Server

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Websense server port</td>
<td>This is the TCP port that receives requests for content filtering according to the Websense protocol.</td>
<td>15868</td>
</tr>
<tr>
<td>Block message server port</td>
<td>If the Websense process blocks a URL, it sends a redirect URL to the user. The redirect URL is configured to print out the blocked page and policy for the user. The Websense process listens on this port to receive the pages blocked, serviced by a thread in the Websense server. This thread sends the blocked page in response to the redirected request.</td>
<td>15871</td>
</tr>
<tr>
<td>Diagnostics server port</td>
<td>The Websense server has an exhaustive set of diagnostics that users can run remotely to diagnose problems in the Websense process. This is the port to which these diagnostics utilities connect.</td>
<td>15869</td>
</tr>
<tr>
<td>Websense configuration server port</td>
<td>This is the port for the Websense Policy Server that the Websense GUI Manager connects to. There is no default entry in the webspense.ini file for this port, and we recommend that you do not modify this default.</td>
<td>55806</td>
</tr>
<tr>
<td>Logon Agent port</td>
<td>This is the port that the Logon Agent, which is running on the Content Engine, listens on for communication from LogonApp.exe, which is running on the Windows client machines. The Logon Agent receives the username, hashed password, and the client machine’s IP address on this port. Support for the Logon Agent was added in the ACNS 5.4.1 software release.</td>
<td>15880</td>
</tr>
</tbody>
</table>

You can configure the first three ports that are listed in Table 11-11 by modifying the eimserver.ini file that resides in the /local1/WebsenseEnterprise/EIM directory on the standalone Content Engine. The Websense server must be restarted so it can pick up the newly configured ports.

You can modify the ports by exporting a copy of the eimserver.ini file using FTP from the /local1/WebsenseEnterprise/EIM directory on the Content Engine, modifying the file, deleting the eimserver.ini file on the Content Engine, and then sending back the modified file to the Content Engine using FTP.

Note: The Websense server needs to be disabled and then reenabled to pick up newly configured ports. To disable the local Websense server, use the no websense-server enable global configuration command. Also make sure that you use the url-filter http websense server global configuration command to point the Websense client to the correct Websense server port. For more information about the url-filter http websense server command, see the Cisco ACNS Software Command Reference, Release 5.5 publication.
Websense Issues When Upgrading to ACNS 5.4.x Software

If you upgrade from the ACNS 5.3.x software to the ACNS 5.4.x software, support for the RADIUS and eDirectory configurations will no longer be provided by the Websense binaries. Consequently, the global configuration commands that were used to configure the RADIUS and eDirectory agents are deprecated in the ACNS 5.4.1 software release.

If you have used the CLI configuration commands to configure the RADIUS and eDirectory agents on a Content Engine that is running the ACNS 5.3.x software, and you upgrade from the ACNS 5.3.x software to the ACNS 5.4.x software, the configuration for the RADIUS and eDirectory agents that were stored in the wsradius.ini and wsendire.ini files will be retained. However, any configuration changes that are made to the RADIUS and eDirectory agents through the Websense GUI Manager will not be reflected in these two .ini files.

The CLI commands that are used to activate the RADIUS and eDirectory agents (the `websense-server service radius-agent activate` and `websense-server service edirectory-agent activate` global configuration commands) and to specify the eDirectory administrative password (the `websense-server service edirectory-agent edir-server administrative-passwd` `password` global configuration command) have been retained in the ACNS 5.4.x software.

If you are upgrading from the ACNS 5.2.x software to the ACNS 5.4.x software, you do not have to make any changes to the previous Websense configuration because the RADIUS and eDirectory Agents were not supported in the ACNS 5.2.x software release.

Websense Issues When Downgrading to an Earlier Version of the ACNS Software

If you downgrade from the ACNS 5.4.x software to the ACNS 5.3. software, the local WebsenseEnterprise directory and other related Websense files are removed. All previously existing internal configuration files used by Websense are deleted, and all changes that were made before the downgrade are lost.

The following error message is displayed to notify you about this Websense downgrade issue:

```text
WARNING:
Websense does not support downgrade
Hence removing /local/local1/WebsenseEnterprise
Websense will stop working after copy ftp install
```

In previous releases, the software did not generate an error message indicating that the WebsenseEnterprise directory had been removed.

When the Content Engine reloads after the downgrade, Websense is reinstalled, and the internal configuration files are recreated. Content Engine Websense server configurations (such as `websense-server service policy local activate` and `websense-server service eim activate`) are reinitiated and they are stored in the startup configuration before the downgrade.

If you downgrade from the ACNS 5.4.x software to the ACNS 5.3. software, the startup configuration will remain intact. All previously existing configuration files are cleaned up and a fresh install is performed. All changes that were made earlier are lost. After rebooting the Content Engine, the CLI configuration is taken from the startup configuration.

If you downgrade from the ACNS 5.4.x software to the ACNS 5.2.x or 5.1.x software, the configured CLI commands and the Websense configuration files on the Content Engine remain intact.
If the local (internal) Websense server is enabled on the Content Engine and you downgrade from the ACNS 5.2.x software to either the ACNS 5.0 software or the ACNS 5.1 software, the WebsenseEnterprise directory is removed from the Content Engine and the local Websense server stops working. To avoid this problem when downgrading from the ACNS 5.2.x software or later to either the ACNS 5.1 software or the ACNS 5.0 software, follow these steps:

**Step 1** Disable the local (internal) Websense server on the Content Engine.

**Step 2** Deactivate the Websense services on the Content Engine.

**Step 3** Install the ACNS 5.1 software or the ACNS 5.0 software downgrade image on the Content Engine.

---

**Example of Configuring Websense Server Failover and URL Filtering**

In the following example, the Content Engine is acting as the HTTP proxy for URL filtering. The Content Engine is first configured to use either the local or the remote Policy Server, and then the local Websense server services (the local EIM Server, the local User Service, and the local Network Agent) are activated on the Content Engine.

The Content Engine next is configured to use the local (internal) Websense server as its primary Websense server and an external Websense server as the secondary Websense server. If the primary Websense server is unavailable, the Content Engine sends the filtering requests to this secondary server.

After allow mode is reenabled on the Content Engine, URL filtering is enabled on the Content Engine. The Websense manager GUI is used to configure the default policy for the local and the remote Websense servers, and then the HTTP proxy is enabled on the Content Engine.

To configure Websense server failover and URL filtering, follow these steps:

**Step 1** Specify whether the local or remote Websense Policy Server is to be used to activate the individual Websense services on the Content Engine.

- To use the local Policy Server, activate the local Policy Server on the Content Engine, as follows:
  
  ```
  ContentEngine(config)# websense-server service policy local activate
  ```

- To use a remote Policy Server, configure the necessary information about the remote Policy Server (for example, its hostname or IP address, and its port number) on the Content Engine, as follows:
  
  ```
  ContentEngine(config)# websense-server service policy remote host {hostname|IP address} [port policy-server-port]
  ```

  where:

  - `hostname` or `IP address` is the hostname or IP address of the remote Policy Server.
  - The port number is optional. The default port number is 55806.

Either the local or the remote Policy Server must be running before you can activate any of the services of the local Websense server (the local EIM Server, the local User Service, and the local Network Agent, the local RADIUS Agent, and the local User Service) on the Content Engine. Local and remote Policy Server configuration are mutually exclusive.

**Step 2** Activate the local EIM Server on the Content Engine.

```
ContentEngine(config)# websense-server service eim activate
```
Step 3  Activate the local User Service on the Content Engine.
ContentEngine(config)# websense-server service user activate

Step 4  Activate the local Network Agent on the Content Engine.
ContentEngine(config)# websense-server service network-agent activate

Step 5  Activate the local eDirectory Agent on the Content Engine.
ContentEngine(config)# websense-server service edir-agent activate

Step 6  Activate the local RADIUS Agent on the Content Engine.
ContentEngine(config)# websense-server service radius-agent activate

Step 7  Activate the local Logon Agent on the Content Engine.
ContentEngine(config)# websense-server service logon-agent activate

Step 8  Enable all of the services of the local Websense server (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service) that have been activated on the Content Engine.
ContentEngine(config)# websense-server enable

**Note**  By default, the local Websense server, which consists of the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service, is disabled on a Content Engine. If you are using the local Websense server with a cluster of standalone Content Engines, make sure that you enable the local Websense server on each standalone Content Engine (that is, enter the `websense-server enable` global configuration command on each Content Engine in the Content Engine cluster).

Step 9  Configure the Content Engine to use the local Websense server as the primary Websense server by using the `url-filter http websense server local` global configuration command.

**Note**  You can use the `url-filter http websense server` global configuration command to configure different settings (for example, the timeout, port number, and the number of connections) for the primary and secondary Websense servers. By default, the Content Engine (that is acting as the HTTP proxy) sends filtering requests to the Websense server on port 15868, waits 20 seconds for a response from the Websense server before timing out the connection, and establishes 40 persistent connections per CPU.

In this example, the Content Engine (that is acting as the HTTP proxy) sends filtering requests to the local Websense server on port 4005, waits 60 seconds for a response from the local Websense server before timing out the connection, and establishes 90 persistent connections to this local Websense server. Because the local Websense server is configured first, it is designated the primary Websense server for the Content Engine.

ContentEngine(config)# url-filter http websense server local port 4005 timeout 60 connections 90

**Note**  The IP address of the local Websense server cannot be configured and is set at 127.0.0.1.
Chapter 11  Configuring Content Preloading and URL Filtering on Standalone Content Engines

Step 10  Configure the Content Engine to use an external Websense server as the secondary Websense server by using the `url-filter http websense server` global configuration command.

Because the local Websense server is already the primary Websense server, you must specify an external Websense server as the secondary Websense server.

In this example, the external Websense server with an IP address of 172.18.22.10 is configured as the secondary Websense server. If the local Websense server is unavailable, the Content Engine will send the requests to this secondary Websense server on port 4006, will wait up to 90 seconds for a response from this server before timing out the connection, and will establish 90 persistent connections per CPU.

```
ContentEngine(config)# url-filter http websense server 172.18.22.10 port 4006 timeout 90
```

Step 11  By default, allow mode is enabled. To reenable allow mode, enter the following command:

```
ContentEngine(config)# no url-filter http websense allowmode enable
```

If the primary Websense server is unavailable, then the Content Engine sends the requests to the specified secondary Websense server. If both the primary and the secondary Websense servers are unavailable, then the requests are sent to allow mode.

- When allow mode is enabled, the Content Engine allows all HTTP traffic to continue through it (it proceeds with normal traffic processing) even if it fails to receive responses from the Websense server.
- When allow mode is disabled, the Content Engine blocks all HTTP traffic that is served through it if it fails to receive responses from the Websense server.

You can configure the `allowmode` option with or without the Websense server being enabled; it is independent of the Websense server configuration. The Content Engine adopts the new configuration for `allowmode` if Websense URL filtering is already being used.

Step 12  Enable URL filtering on the Content Engine.

```
ContentEngine(config)# url-filter http websense enable
```

Step 13  Configure the default policy using the Websense Manager GUI. This step should be performed for both the local and the remote Websense server.

a. Use the Websense Manager GUI to add a Policy Server.
   - Right-click the left pane of the Websense Manager main window.
   - Choose Add Policy Server.
   - In the displayed dialog box, enter the IP address of the Content Engine that is running the local (internal) Websense server.

b. Connect to the Websense Policy Server that is running on the Content Engine.
   - In the left pane, double-click on the Policy Server (this could be the Content Engine IP address, for example).
   - Enter the username and password, and then click OK.
c. Use the Websense Manager GUI to configure a Websense policy.
   - Use the Websense Manager GUI to connect to the Websense Policy Server.
   - In the left pane, double-click **Filter Definition** and then **Policies**.
   - Choose **Global**.
   - In the right pane, click the **Edit** button.
   - In the displayed dialog box, apply such category sets as the default settings, basic settings, always block, and never block. The default policy is global, and the default category set is the default settings.

   **Note**
   Clicking the **Save Changes** button from the Websense Enterprise Manager window does not save the Websense configuration modifications across device reboots. You need to use the **write memory** command to save the Websense configuration changes across reboots. For more information about how to use the Websense Manager GUI, go to the following website: http://www.websense.com.

**Step 14** Configure the HTTP proxy on the Content Engine.

```
ContentEngine(config)# http proxy incoming 8080
```

**Step 15** Display statistics for both the primary and the secondary Websense servers.

```
ContentEngine# show statistics url-filter http websense
```

### Configuring URL Filtering with a Local Websense Server

To configure the Content Engine to use the local (internal) Websense server for URL filtering, you must perform these tasks:

1. Use the local or remote Policy Server to activate the local Websense server services (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service) on the Content Engine.

2. Enable the local Websense server on the Content Engine. (By default, it is disabled.)

3. Configure the Content Engine to use the local Websense server for URL filtering of HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests). If the Network Agent is configured, other protocols can be filtered as well.

In the ACNS 5.2.1 software and later releases, you can configure up to two Websense servers for failover purposes. One of these Websense servers can be the local Websense server. The order in which you configure the Websense servers determines which server is the primary server. The first configured Websense server is automatically designated the primary Websense serve, and the second configured server becomes the secondary Websense server. For a list of supported configurations, see Table 11-8.

In the ACNS 5.3.1 software and later releases, you can activate any combination of the local Websense server services (which are listed in Table 11-9). If the local Policy Server is not activated on the Content Engine, you must point to a valid external Policy Server when activating the other local Websense server services (the local EIM Server, the local Network Agent, the local RADIUS Agent, the local eDirectory Agent, and the local User Service). For more information on this topic, see the “Configuring Websense URL Filtering with External Websense Servers” section on page 11-36.
Chapter 11 Configuring Content Preloading and URL Filtering on Standalone Content Engines

Configuring URL Filtering on Standalone Content Engines

The ACNS 5.0.3 to 5.1.x software has the local Websense server. Because the activation of the individual services of the Websense server in these software releases is not optional, by default the Websense server services are activated on the Content Engine as follows:

- When you upgrade to the ACNS 5.4.x software from the ACNS 5.3.x software releases, only the local Policy Server, the local EIM Server, the local eDirectory Agent, the local RADIUS agent, and the local User Service will be activated. (The local Logon Agent is not activated upon upgrade to the ACNS 5.4.1 software release).

- When you upgrade to the ACNS 5.3.x software from the ACNS 5.0.3, 5.1.x, or 5.2.x software releases, only the local Policy Server, the local EIM Server, and the local User Service will be activated. (The local eDirectory Agent and the local RADIUS Agent are not activated upon upgrade to the ACNS 5.3.1 software release).

- When you upgrade to the ACNS 5.2.1 software and later releases from the ACNS 5.0.3 or 5.1.x software releases, the following three local Websense server services are activated on the Content Engine: the local Policy Server, the local EIM Server, and the local User Service.

To configure the Content Engine to use the local (internal) Websense server for URL filtering, follow these steps:

**Step 1** Specify whether the local or remote Policy Server is to be used to activate the individual services of the local Websense server on the Content Engine.

- To use the local Policy Server, activate the local Policy Server on the Content Engine.

  ContentEngine(config)# websense-server service policy local activate

- To use a remote Policy Server, configure the necessary information about the remote Policy Server (for example, its hostname or IP address, and its port number) on the Content Engine.

  ContentEngine(config)# websense-server service policy remote host {hostname|IP address} [port policy-server-port]

  where:
  - **hostname** or **IP address** is the hostname or IP address of the remote Policy Server.
  - The port number is optional. The default port number is 55806.

**Note** Either the local or the remote Policy Server must be running before you can activate any of the services of the local Websense server (the local EIM Server, the local Network Agent, the local RADIUS Agent, the local eDirectory Agent, the local Logon Agent, and the local User Service) on the Content Engine. Local and remote Policy Server configuration are mutually exclusive.

**Step 2** Activate the local EIM Server on the Content Engine.

  ContentEngine(config)# websense-server service eim activate

**Step 3** Activate the local User Service on the Content Engine.

  ContentEngine(config)# websense-server service user activate

**Step 4** Activate the local Network Agent on the Content Engine.

  ContentEngine(config)# websense-server service network-agent activate
Chapter 11 Configuring Content Preloading and URL Filtering on Standalone Content Engines

Configuring URL Filtering on Standalone Content Engines

Step 5 Configure the settings for the local eDirectory agent, which will be running on the Content Engine, through the Websense Manager GUI (Version 5.5, or Version 6.1 or later).

Step 6 Configure the settings for the local RADIUS Agent, which will be running on the Content Engine, through the Websense Manager GUI (Version 5.5, or Version 6.1 or later).

Step 7 Activate the local eDirectory Agent on the Content Engine.

```
ContentEngine(config)# websense-server service edir-agent activate
```

Step 8 Activate the local RADIUS Agent on the Content Engine.

```
ContentEngine(config)# websense-server service radius-agent activate
```

Step 9 Enable all of the services of the local Websense server (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service) that have been activated on the Content Engine.

```
ContentEngine(config)# websense-server enable
```

Note By default, the local Websense server, which consists of the local EIM Server, the local Network Agent, and the local User Service, is disabled on a Content Engine. The IP address of the local Websense server cannot be configured and is set at 127.0.0.1. If you are using the local Websense server with a cluster of standalone Content Engines, make sure that you enable the local Websense server on each standalone Content Engine (for example, enter the `websense-server enable` global configuration command on each Content Engine in the Content Engine cluster).

Step 10 Configure the Content Engine to use the local Websense server for URL filtering of HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests). See Table B-4 for the kinds of URL filtering that are supported for different protocols with a Websense server.

```
ContentEngine(config)# url-filter http websense server local [port portnumber] [timeout seconds] [connections connections]
```

where:

- **local** specifies that the Content Engine is to use the internal Websense server for URL filtering.
- **port number** specifies the port (1–65535) on which the local Websense server is to listen for HTTP requests to filter. By default, the local Websense server listens on port 15868.
- **seconds** is the number of seconds (0–240) that the Content Engine is to wait for an HTTP response from the internal Websense server before timing out the connection. The default is 20 seconds.
- **connections** is the number of persistent connections (1–250) per CPU (the default is 40 per CPU). Use this option to configure the number of persistent connections to the internal Websense server. Do not change the default number unless you know for certain that a different value is required.

Step 11 Enable Websense URL filtering of HTTP requests.

```
ContentEngine(config)# url-filter http websense enable
```

Step 12 View the current Websense server configuration.

```
ContentEngine# show websense-server
```

For information about deactivating one or more of the services of the local Websense server on the Content Engine, see the “Deactivating Local Websense Server Services on Standalone Content Engines, page 11-37.”
Configuring Websense URL Filtering with External Websense Servers

When configuring a Content Engine to use an external Websense server, you must specify an IP address and port number for that server. That specified IP address and port number must match the IP address of the external Websense server and the port that the external Websense server listens to for filtering requests. Otherwise, the Content Engine will time out all HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) and either block or allow all HTTP traffic based on the allowmode option configuration. By default, allow mode is enabled on the Content Engine. When allow mode is enabled, the Content Engine is permitted to fulfill an HTTP request from a client if the external Websense server does not respond. If allow mode has been disabled, use the url-filter http websense allowmode enable command to re-enable it.

In the ACNS 5.2.1 software and later releases, you can configure up to two Websense servers for failover purposes. The order in which you configure the Websense servers determines which server is the primary server. The first configured Websense server is automatically designated the primary Websense server, and the second configured server becomes the secondary Websense server. For a list of supported Websense server configurations, see Table 11-8.

To configure a standalone Content Engine to use an external Websense server for URL filtering, follow these steps:

Step 1 Specify the necessary information about the external Websense server by using the url-filter http websense server global configuration command.

```
url-filter http websense server { [hostname | ip-address] | [port portnum [timeout seconds [connections connection]]]
```

where:

- **hostname** is the hostname of the external Websense server.
- **IP address** is the IP address of the external Websense server.
- **portnum** is the port number (1–65535) of the external Websense server to which the Content Engine is to send HTTP requests. The default is port 15868.
- **seconds** is the number of seconds (0–240) that the Content Engine is to wait for an HTTP response from the external Websense server before timing out the connection. The default is 20 seconds.
- **connections** is the number of persistent connections (1–250) per CPU (the default is 40 per CPU). Use this option to configure the number of persistent connections to the external Websense server. Do not change the default number unless you know for certain that a different value is required.

The following example shows how to configure a standalone Content Engine to point to an external Websense server that has the IP address 172.18.22.10 and is running on Host A. The Content Engine is configured to send requests to this external Websense server on port 4006, and to wait up to 90 seconds for a response from this server before timing out the connection.

```
ContentEngine(config)# url-filter http websense server 172.18.22.10 port 4006 timeout 90
```

**Note** To use an external Websense server for URL filtering with a cluster of standalone Content Engines, make sure to use the url-filter http websense server global configuration command on each Content Engine in the Content Engine cluster to ensure that all traffic is filtered.
Step 2  If you want to configure a secondary Websense server for failover purposes, take one of the following actions:

- To configure the local (internal) Websense server as the secondary Websense server, use the `url-filter http websense server local` global configuration command. For more information about configuring the local Websense server, see the “Configuring URL Filtering with a Local Websense Server” section on page 11-33.

- To configure an external Websense server that is running on a different host (Host B) than the primary Websense server (Host A), enter another `url-filter http websense server` command. This time, the command should specify the parameters for the secondary Websense server (for example, the IP address, port number, timeout, and number of connections of the Websense server running on Host B.)

Step 3  Enable Websense as the current URL filtering scheme for HTTP on this Content Engine.

```
ContentEngine(config)# url-filter http websense enable
```

*Note*  For information about configuring an external Websense server, go to the following website: http://www.websense.com.

Step 4  View the current Websense server configuration.

```
ContentEngine# show websense-server
```

---

**Deactivating Local Websense Server Services on Standalone Content Engines**

To deactivate one or more of the services of the local Websense server (the local EIM Server, the local Network Agent, the local RADIUS Agent, the local eDirectory Agent, the local User Service, or the local Policy Server) on a standalone Content Engine, follow these steps:

Step 1  Check whether the local Websense server is currently enabled on the Content Engine.

```
ContentEngine# show websense-server
```

Step 2  Disable the local Websense server on the Content Engine, if it is currently enabled.

```
ContentEngine(config)# no websense-server enable
```

Step 3  Deactivate a particular service of the local Websense server by using the `no` form of the `websense-server service` global configuration command.

For example, use the `no websense-server service network-agent activate` command to deactivate the local Network Agent on the Content Engine:

```
ContentEngine(config)# no websense-server service eim activate
ContentEngine(config)# no websense-server service user activate
ContentEngine(config)# no websense-server service network-agent activate
ContentEngine(config)# no websense-server service edir-agent activate
ContentEngine(config)# no websense-server service radius-agent activate
ContentEngine(config)# no websense-server service logon-agent activate
```
There is no required order in which you should deactivate the local EIM Server, the local Network Agent, the local eDirectory Agent, the local RADIUS Agent, the local Logon Agent, or the local User Service on the Content Engine. However, if the Policy Server is running on the Content Engine (that is, the local Policy Server is being used instead of a remote Policy Server), the Policy Server should be the last local Websense service to be deactivated (using the `no websense-server service policy activate` command).

In contrast, if you are using a remote Policy Server, make sure that it is running before you attempt to deactivate the local EIM Server, the local Network Agent, the local eDirectory Agent, the local RADIUS Agent, or the local User Service (using the `no websense-server service service-name activate` command) on the Content Engine.

**Step 4**  
Deactivate the local Policy Server on the Content Engine, if it is being used.

```
ContentEngine(config)# no websense-server service policy
```

**Step 5**  
Unconfigure the remote Policy Server on the Content Engine, if it is being used.

```
ContentEngine(config)# no websense-server service policy remote host
```

### Saving Websense Configuration Files

In the ACNS 5.2.1 software and later releases, the `write memory` command saves modified Websense configuration files (the `eimserver.ini`, `config.xml`, and `websense.ini` files and the `Blockpages` directory) across disk reconfiguration and the ACNS software release upgrades.

You must execute the `write memory` command in order to save the most recent configuration modifications, including `websense.ini` file modifications and Websense URL filtering configuration changes. The `write memory` command enables the changes made from the external Websense Manager GUI to be saved across disk reconfiguration and upgrades (which might erase disk content).

The Websense configurations from the last use of the `write memory` command are retained under the following situations:

- If the `write memory` command is not used before a reboot with a disk reconfiguration or an ACNS software upgrade that erases disk content
- If you are using the `reload` command and did not answer `yes` when asked if you wanted to save the configurations at the reload prompt

If the `write memory` command has never been used before, then default configurations will be applied when the content in the `/local1/WebsenseEnterprise/EIM` directory on the Content Engine is erased.

### Viewing Websense URL Filtering Statistics

To display the status of all HTTP URL filtering schemes presently configured on the standalone Content Engine, enter the `show url-filter http EXEC` command. Use the `show statistics url-filter http websense` EXEC command To display the request-reply statistics for the communication between the Content Engine and the Websense server, enter the `show statistics url-filter http websense` EXEC command. 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 Websense server.

The statistics shown can be cleared using the `clear statistics url-filter http websense` and `clear statistics all` EXEC commands. All the statistics counters are then reset to 0.
Chapter 11 Configuring Content Preloading and URL Filtering on Standalone Content Engines

Configuring Content Preloading and URL Filtering on Standalone Content Engines

Configuring URL Filtering with SmartFilter Software

SmartFilter software running on standalone Content Engines provides employee Internet management (EIM) functionality when used with proxy servers, firewalls, and caching appliances. The SmartFilter filtering capability is available as an add-on service on a Content Engine that is running the ACNS 5.x software. The SmartFilter add-on service is licensed directly through Cisco.

The SmartFilter add-on service 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.

To configure this SmartFilter add-on service, you use an end user management tool called the sfadmin console, and a management server tool called the sfadmin server. You use the sfadmin console to configure the SmartFilter product and then store the configuration on the sfadmin server. The sfadmin server propagates this configuration to the end client Content Engines, to be used by the SmartFilter software that is running on the Content Engines. To enable SmartFilter URL filtering on a standalone Content Engine, use the `url-filter http smartfilter enable` global configuration command. To use SmartFilter URL filtering with a cluster of standalone Content Engines, make sure to enter the `url-filter http smartfilter enable` command on each Content Engine in the cluster to ensure that all traffic is filtered.

Note

The ACNS 5.2.1, 5.3.x, 5.4.1, 5.5.1, and 5.5.3 software releases support SmartFilter software Version 4.0. SmartFilter software Version 4.1 is supported in the ACNS 5.4.3 release. With the SmartFilter software Version 4.1 support, the agent listens directly on port 9014 for block page requests. See Table B-4 for a complete list of protocols that are supported with SmartFilter. See the Release Notes for the most current SmartFilter software version support information for your ACNS release.

When configuring URL filtering with SmartFilter software, remember the following important points:

- When you upgrade or downgrade the Content Engine to a different release of the ACNS software, if there is a difference in the SmartFilter plug-in version, the SmartFilter database and configuration files are deleted and default configurations are loaded. This change occurs because the configuration details might be changed with each new version of the SmartFilter software. After each upgrade or downgrade of the SmartFilter plug-in, a fresh database has to be downloaded from the SmartFilter Administration Console to the Content Engine.

- If the Content Engine is deployed in a natted environment, you must use the `external-ip external-ip-address` global configuration command to configure the Content Engine’s external IP address, otherwise the Advanced block page feature may not work properly.

- The Smartfilter software and the cache process on the Content Engine will be restarted in the following situations:
  - If you configure the Content Engine’s external IP address when the Smartfilter software is running on the Content Engine.
  - If the Content Engine’s external IP address is not configured and you change the Content Engine’s interface IP address when the Smartfilter software is running on the Content Engine.

- Port 9014 is reserved for Smartfilter Advanced block pages.
About the SmartFilter Control List

A SmartFilter Control List categorizes 2 million websites into content groups. There are 30 predefined SmartFilter Control List categories that encompass a wide variety of material. Some categories are focused on reducing legal liability of a company. These 30 categories are set to Deny in the default SmartFilter software policy. Some categories contain such sites as MP3 sites (sites with content that consumes excessive bandwidth). The remainder of these 30 categories are considered unproductive or inappropriate for business or educational environments.

The SmartFilter software also provides ten user-defined categories that allow you to further tailor access by defining and filtering sites that are not included in the SmartFilter Control List. Additionally, you can exempt any site that you would like specific groups or individuals to access quickly and easily. You can use the SmartFilter Administration Console to define a SmartFilter Control List download schedule. The Download Setup window tracks the download site, your username, and your password. If you do not download an updated SmartFilter Control List at least monthly, the SmartFilter software considers the Control List expired, and invokes the action that you specified in the SmartFilter License window.

About the Temporary User Override Feature

In the ACNS 5.4.1 software release, the temporary user override was a new feature that is available with the SmartFilter software Version 4.1. This feature allows certain users to override the filtering process that is currently being applied to their user group. You must configure this feature through the SmartFilter Administrator Console.

To use the temporary user override feature, follow these steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install the SmartFilter authentication server software on the machine that is running the SmartFilter Administrator Console or on a different machine.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>To obtain a copy of the Smartfilter authentication server software, go to the following website: <a href="http://www.securecomputing.com">http://www.securecomputing.com</a>.</td>
</tr>
<tr>
<td>2</td>
<td>From the SmartFilter Administrator Console, add the authentication server.</td>
</tr>
<tr>
<td>3</td>
<td>From the SmartFilter Administrator Console, add the users the authentication server.</td>
</tr>
<tr>
<td>4</td>
<td>Send the changes to the authentication server.</td>
</tr>
<tr>
<td>5</td>
<td>From the SmartFilter Administrator Console, select the Content Engine and add the configured authentication server to the Content Engine’s list of authentication servers.</td>
</tr>
<tr>
<td>6</td>
<td>From the SmartFilter Administrator Console, add the users who should be allowed to override the filtering process in the overrides for the Content Engine.</td>
</tr>
<tr>
<td>7</td>
<td>Send the changes to the Content Engine.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>For more information about configuring the SmartFilter software, go to the following website: <a href="http://www.securecomputing.com">http://www.securecomputing.com</a>.</td>
</tr>
</tbody>
</table>
Configuring Content Engines to Bypass URL Filtering for Specific HTTP and HTTPS Requests

In the ACNS 5.2.3 software and later releases, you can configure a Content Engine to bypass URL filtering for certain HTTP and HTTPS requests. This feature is supported for local list URL filtering (good and bad site lists), as well as Websense, SmartFilter, or N2H2 URL filtering.

For example, if you enable local URL filtering on the Content Engine and enable the bad sites deny feature (for example, the badfile.txt file contains the URLs that should be blocked), and the rule no-url-filtering action is a hit (a match), the Content Engine bypasses the URL filtering for that particular request; otherwise, it proceeds with URL filtering and blocks the URL request.

To configure this feature on a standalone Content Engine, use the rule action no-url-filtering global configuration command. For more information on this topic, see the “Example of no-url-filtering Action” section on page 13-14.

Displaying the Current URL Filtering Configurations

To display the URL filtering configurations for a standalone Content Engine, use the show url-filter EXEC commands:

```
ContentEngine# show url-filter http
ContentEngine# show url-filter rtsp
ContentEngine# show url-filter wmt
```

Caution

If the size of the local list file becomes too large, it can adversely affect proxy performance, because the local list file is loaded into memory when local list filtering is enabled. If the file size is larger than 5 MB, a warning message appears, but the ACNS software does not enforce size limits for the local list file. It is your responsibility to track the local list file size and ensure that it does not become so large that it degrades performance.

Displaying URL Filtering Statistics

To display statistics for the various URL filtering schemes that are configured on a standalone Content Engine, use the show statistics url-filter EXEC commands.

```
ContentEngine# show statistics url-filter ?
  http  Display URL-filter for http and mms over http statistics
  rtsp  Display URL-filter for rtsp statistics for real proxy, real server and cisco streaming engine
  wmt   Display URL-filter for wmt statistics for rtsp requests
```

To display local list URL filtering statistics for WMT requests (RTSP requests from Windows Media 9 players), enter the show statistics url-filter wmt local-list EXEC command.

Note

For WMT requests, local list files is the only supported URL filtering scheme.

As the sample command output shows, the number of allowed WMT requests, blocked WMT requests, and WMT requests not filtered by the local list URL filtering are displayed:

```
ContentEngine# show statistics url-filter wmt local-list
```
Local List URL filtering statistics:
Requests Allowed = 25
Requests Blocked = 30
Requests not filtered = 5

To display RealMedia request statistics for a standalone Content Engine, enter the `show statistics url-filter rtsp local-list` EXEC command. RealMedia requests are handled by the RealProxy server that is running on the standalone Content Engine.

```
ContentEngine# show statistics url-filter rtsp?
local-list Display local-list URL-filter statistics
```

**Note**
For RealMedia requests, local list files is the only supported URL filtering scheme. For registered Content Engines (that is Content Engines that are registered with a Content Distribution Manager as opposed to standalone Content Engines that are initially not registered with a Content Distribution Manager if there is one in the ACNS network), the command output from the `show statistics url-filter rtsp local-list` EXEC command will also include statistics that the RealSubscriber and Cisco Streaming Engine have served to clients if these two backend RTSP servers have been enabled on the registered Content Engine.

As the sample command output shows, the number of allowed requests, blocked requests, and the number of requests not filtered by the local list URL filtering are displayed:

```
ContentEngine# show statistics url-filter rtsp local-list
Local List URL filtering statistics:
Requests Allowed = 15
Requests Blocked = 10
Requests not filtered = 2
```

To display URL filtering statistics for HTTP requests, enter the `show statistics url-filter http` EXEC command. With URL filtering for HTTP requests, local list files and URL filtering through third-party software (for example, N2H2 and Websense software) are supported.

```
ContentEngine# show statistics url-filter http ?
local-list Display local-list URL-filter statistics
N2H2 Display N2H2 URL-filter statistics
websense Display websense URL-filter statistics
```

### Clearing URL Filtering Statistics

To clear URL filtering statistics on standalone Content Engines, use the `clear statistics url-filter` EXEC commands.

```
ContentEngine# clear statistics url-filter ?
http Clear URL-filter for http statistics
rtsp Clear URL-filter for rtsp statistics
wmt Clear URL-filter for wmt statistics
```

For example, clear the WMT URL filtering statistics on a standalone Content Engine, as follows:

```
ContentEngine# clear statistics url-filter wmt local-list
```
Configuring ICAP on Standalone Content Engines

This chapter describes how to configure the Internet Content Adaptation Protocol (ICAP) on standalone Content Engines. The ACNS 5.2.1 software and later releases support ICAP for HTTP requests. The ACNS 5.4.1 software and later releases support ICAP for FTP-over-HTTP requests. Support for native FTP requests is not supported.

This chapter includes the following topics:

- Overview of ICAP, page 12-1
- Example of Configuring ICAP Services on a Content Engine, page 12-4
- Configuring ICAP for Standalone Content Engines, page 12-5
- Configuring Logging of ICAP Exchanges, page 12-9
- Displaying Information About an ICAP Configuration, page 12-10
- Displaying Statistics for ICAP Services, page 12-10

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication.

For information about how to configure ICAP for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

Overview of ICAP

ICAP is an open standards protocol for content adaptation, typically at the network edge. Content adaptation includes virus scanning, content translation, content filtering, content insertion, and other ways of improving the value of content to end users. ICAP specifies how a Content Engine, acting as an HTTP proxy server, can communicate with an external device that is acting as an ICAP server, which filters and adapts the requested content.

ICAP provides two content-processing modes for HTTP services. These modes define the transactions that can occur between a Content Engine acting as an ICAP client and an ICAP server. The two modes are as follows:

- Request modification (reqmod)—Allows modification of requests as they are sent from the Content Engine to the ICAP server on their way to the origin server. The ICAP server can modify these requests depending on the services requested.
Overview of ICAP

- Response modification (respmod)—Allows modification of requests after they return from the origin server. The ICAP server only acts on requested objects after they return from the origin server.

**Note**
The ACNS 5.4.1 software and later releases also support ICAP for FTP-over-HTTP requests. Support for native FTP requests is not supported.

About ICAP Services

An ICAP service is a collection of attributes that define the service and one or more ICAP servers that provide the ICAP services. You can configure a maximum of ten ICAP services per Content Engine, with an upper limit of five ICAP servers per ICAP service. Also, you can choose to apply ICAP services on all HTTP requests processed by the Content Engine or apply ICAP processing only to requests that match the Rules Template.

**Tip**
To set the type of load balancing to use among a cluster of ICAP servers, use the `icap service load balancing` global configuration command.

About ICAP Services and Vectoring Points

The point at which ICAP services are applied to content is called the *vectoring point*, specified using the `vector-point` option. The following three vectoring points are supported:

- **Client request vectoring point (reqmod-postcache)**—The ICAP server performs one of the following actions in response to the client request:
  - Terminates the connection
  - Sends a modified error response
  - Searches the cache using the URL in the request
  - Searches the cache using a modified URL
  - Modifies the request header or request body in the case of a cache miss

- **Cache miss vectoring point (reqmod-precache)**—The ICAP server performs one of the following actions before forwarding the request to the origin server:
  - Terminates the connection
  - Sends a modified error response
  - Sends the request to the origin server using the original URL
  - Sends the request to the origin server using an alternative URL
  - Modifies the request header or request body

- **Server response vectoring point (respmod-precache)**—The ICAP server performs one of the following actions after receiving the response from the origin server:
  - Returns the response to the client
  - Modifies the request header or request body
  - Caches the response using the original URL
  - Caches the response using an alternative URL
Overview of ICAP

Note

Different ICAP services assigned to the same vectoring point can use different load-balancing options.

The following commands show a typical configuration for a virus-scanning service that requires processing on two vectoring points: `reqmod-precache` and `respmod-precache`:

```
ContentEngine(config)# icap apply all
ContentEngine(config)# icap service trend-reqmod
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point reqmod-precache
ContentEngine(config-icap-service)# server icap://172.19.227.150/REQ-Service
ContentEngine(config-icap-service)# exit
ContentEngine(config-icap-service)# icap service trend-respmod
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point respmod-precache
ContentEngine(config-icap-service)# server icap://172.19.227.150/interSCAN
ContentEngine(config-icap-service)# exit
```

If an ICAP vendor supports the same service name for more than one vectoring point, you can configure a single service and add the supported vectoring points, as in the following example:

```
ContentEngine(config)# icap service myicap-service
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point reqmod-precache
ContentEngine(config-icap-service)# vector-point respmod-precache
ContentEngine(config-icap-service)# server icap://172.19.227.150/icap-service-name
ContentEngine(config-icap-service)# exit
ContentEngine(config)#
```

About ICAP Performance

With the response modification (respmod) vectoring point, which is used by virus-scanning ICAP vendors, the performance of the Content Engine model CE-7305 will be 300 transactions per second.

With the request modification (reqmod)-precache vectoring point, which is used by URL filtering ICAP vendors, the performance of the Content Engine model CE-7305 will drop 20 percent from the rated performance.

Note

The performance of the Content Engine will be limited by the performance of the ICAP server.

ACNS Software and ICAP Services Interoperability Notes

This section provides information about using ICAP processing services with the ACNS software.

ICAP Vendors Supported

The following is a complete list of the ICAP vendors that have been certified to interoperate with the Content Engine:

- TrendMicro for reqmod and respmod
- Symantec for respmod
Maximum File Size Supported

For ACNS 5.4.x software and later, the maximum file size that is supported in the ACNS software is 2 GB. Files that exceed this size limit are not supported for ICAP processing.

For releases prior to ACNS 5.4.x software, the maximum file size that is supported in the ACNS software in pass-through mode is 2 GB. Files that exceed this size limit are not supported for ICAP processing.

Example of Configuring ICAP Services on a Content Engine

The following procedure is a sample of how ICAP services might be defined and enabled on a standalone Content Engine:

**Step 1** Specify which ICAP services should be performed on which requests that are received by the Content Engine by using the `icap apply {all | rules-template}` global configuration command. For example:

- Instruct the Content Engine to run only the ICAP services that match the rules action `use-icap-service`.
  
  ```
  ContentEngine(config)# icap apply rules-template
  ```

- Instruct the Content Engine to run all of the ICAP services on all of the HTTP requests that it receives.
  
  ```
  ContentEngine(config)# icap apply all
  ```

**Step 2** Enable ICAP-related transaction logging, which is available in the `local1/logs/icap/` directory.

```
ContentEngine(config)# icap logging enable
```

**Step 3** Configure and enable various ICAP services on this Content Engine.

- Define the ICAP service rules for this Content Engine by using the `rule` global configuration command.
  
  ```
  ContentEngine(config)# rule enable
  ContentEngine(config)# rule action use-icap-service trend-reqmod pattern-list 1 protocol all
  ContentEngine(config)# rule action use-icap-service trend-respmod pattern-list 1 protocol all
  ContentEngine(config)# rule pattern-list 1 domain "!(.*cisco\.com|.*datek\.com)"
  ```
Tip
By default, requests from streaming media clients are bypassed for ICAP processing. You can, however, use the `icap bypass streaming-media` global configuration command to enable the bypass feature for streaming media if necessary.

Note
For more information about ICAP services, see the “About ICAP Services and Vectoring Points” section on page 12-2.

Configuring ICAP for Standalone Content Engines

To configure ICAP on a standalone Content Engine, you must use the Content Engine CLI. The following sections describe the tasks you must complete:

- Configuring ICAP Settings for Standalone Content Engines, page 12-5
- Configuring ICAP Services on Standalone Content Engines, page 12-7
- Configuring an ICAP Server for Standalone Content Engines, page 12-8

Configuring ICAP Settings for Standalone Content Engines

The Content Engine CLI must be used to configure ICAP settings on a standalone Content Engine. (The Content Engine GUI does not currently support the configuration of ICAP settings.)

When using the `icap` global configuration command to set ICAP parameters on a standalone Content Engine, keep the following important points in mind:

- To specify the ICAP extension headers that are passed to the ICAP server during the session negotiation between the Content Engine and the ICAP server, use the `icap append-x-headers` global configuration command.

- In the ACNS 5.1 software and later releases, you can configure the Content Engine to append the client and server IP address headers to the request that is passed to the ICAP server. This capability allows you to use your ICAP server to perform URL filtering based on the client IP address and server IP address. To enable this capability, you must use the `icap append-x-headers x-client-ip` and `icap append-x-headers x-server-ip` command options.

- In the ACNS 5.2 software and later releases, you can configure the Content Engine to append the username and group name headers to the request that is passed to the ICAP server. This capability allows you to use your ICAP server to perform URL filtering based on username and group name. Use the following two command options of the `icap append-x-headers` global configuration command, as follows:

  - Specify the `x-authenticated-user` option to allow the username information to be passed to the ICAP server for global services. This option is disabled by default. When this option is enabled (the `icap append-x-headers x-authenticated-user` option), the x-authenticated-user information is inserted into the ICAP request to the ICAP server.
Specify the `x-authenticated-groups` option to allow the group name information to be passed to the ICAP server for global services. This option is disabled by default. When this option is enabled (the `icap append-x-headers x-authenticated-groups` option), the x-authenticated-groups information is inserted into the ICAP request to the ICAP server.

The currently supported authentication schemes include LDAP, NTLM, RADIUS, and TACACS+.

- Requests from streaming media clients are by default bypassed for ICAP processing.

To force strict rechecking of the cached content every time the ISTag changes, use the `icap bypass streaming-media` command. ISTag is a field in the HTTP response header that allows ICAP servers to send a service-specific cookie to an ICAP client, representing the current state of the service. The ISTag may change as a result of an update to the server version, to a virus-pattern-file, or to the policy.

- In the ACNS 5.5.1 software and later releases, you can configure the Content Engine ICAP connection timeout.

To configure a timeout value for ICAP connections, use the `icap connection-timeout minutes` global configuration command.

Table 12-1 describes the `icap` global configuration command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>append-x-headers</td>
<td>Appends x-headers during ICAP protocol handshake. Disabled by default. Can have multiple entries for various x-headers to be appended.</td>
</tr>
<tr>
<td>x-client-ip</td>
<td>Appends x-client-IP headers to the request that is sent to the ICAP server. Disabled by default.</td>
</tr>
<tr>
<td>x-server-ip</td>
<td>Appends x-server-IP headers to the request that is sent to the ICAP server. Disabled by default.</td>
</tr>
<tr>
<td>x-authenticated-user</td>
<td>Appends x-authenticated-user headers to the request that is sent to the ICAP server. Disabled by default.</td>
</tr>
<tr>
<td>x-authenticated-groups</td>
<td>Appends x-authenticated-groups headers to the request that is sent to the ICAP server. Disabled by default.</td>
</tr>
<tr>
<td>apply</td>
<td>Enables ICAP processing for HTTP and FTP-over-HTTP requests.</td>
</tr>
<tr>
<td>all</td>
<td>Enables ICAP processing for all HTTP and FTP-over-HTTP requests.</td>
</tr>
<tr>
<td>rules-template</td>
<td>Enables ICAP processing for HTTP and FTP-over HTTP requests that match the Rules Template using the rule action use-icap-service global configuration command.</td>
</tr>
<tr>
<td>bypass</td>
<td>Enables bypassing of certain requested content.</td>
</tr>
<tr>
<td>streaming-media</td>
<td>Enables bypassing of streaming media.</td>
</tr>
<tr>
<td>connection-timeout</td>
<td>Sets the connection timeout for an ICAP transaction.</td>
</tr>
<tr>
<td>minutes</td>
<td>Number of minutes before the connection times out. The range is 1–480 minutes. The default is 20 minutes.</td>
</tr>
<tr>
<td>logging</td>
<td>Specifies logging-related options when ICAP services are used.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables logging when ICAP services are used.</td>
</tr>
<tr>
<td>format</td>
<td>Specifies the logging format.</td>
</tr>
<tr>
<td>custom</td>
<td>Specifies a customized format for logging.</td>
</tr>
</tbody>
</table>
Configuring ICAP Services on Standalone Content Engines

In the ACNS 5.1 software and later releases, three vectoring points are supported to enable content adaptation, as described earlier in the “About ICAP Services and Vectoring Points” section on page 12-2. ICAP servers configured at various vectoring points may become overloaded with HTTP requests, especially the request modification precache vectoring points because all requests pass through this point. Therefore, a cluster of ICAP servers (a load-balanced collection of ICAP servers) is made available for configuration. At a particular vectoring point, you can choose to load balance requests among the ICAP cluster of servers based on various parameters such as weighted load, client IP and server IP address-based hash, or round-robin format.

More than one ICAP service can be associated with a vectoring point. An ICAP service configured at a vectoring point can have only one load-balancing scheme, regardless of the number of servers. However, multiple ICAP services configured at one or all of the vectoring points can have different load-balancing schemes.

Tip

If you click the Aggregate Settings radio button the ICAP Services for Content Engine window, the ICAP services that have been previously configured for device groups to which the Content Engine belongs cannot be modified or deleted. In other words, you can only view the ICAP services created for the device groups.

To configure ICAP services for a standalone Content Engine, use the icap service global configuration command:

```
```

To configure a specific ICAP service, enter ICAP configuration mode by using the icap service service-id command.

Replace service-id with a name of your choice for the current ICAP service. When you enter the icap service command and provide a name for the ICAP service, the system displays the ICAP service configuration prompt:

```
ContentEngine(config-icap-service)#
```

Within ICAP service configuration mode, all commands that you enter apply to the current ICAP service. Table 12-2 describes the icap service command parameters for configuring an ICAP service on a standalone Content Engine.

**Table 12-2 Parameters for the icap service Command**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service-id</td>
<td>Specifies a name of your choice for the current ICAP service.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables ICAP services.</td>
</tr>
</tbody>
</table>
Table 12-2 Parameters for the icap service Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error-handling</td>
<td>Specifies error-handling options when an ICAP service is used.</td>
</tr>
<tr>
<td>bypass</td>
<td>Bypasses this service when an error occurs with this service.</td>
</tr>
<tr>
<td>return-error</td>
<td>Returns an error message to the client and ends the request.</td>
</tr>
<tr>
<td>load-balancing</td>
<td>Specifies a load-balancing option for this service. See Table 12-3 for a list of these load-balancing options.</td>
</tr>
<tr>
<td>client-ip-hash</td>
<td>Allows for load-balancing among ICAP servers using the client IP address.</td>
</tr>
<tr>
<td>round-robin</td>
<td>Allows for round-robin load balancing among ICAP servers.</td>
</tr>
<tr>
<td>server-ip-hash</td>
<td>Allows for load balancing using the ICAP server IP address.</td>
</tr>
<tr>
<td>weighted-load</td>
<td>Allows for load balancing using a weight scheme that specifies weight on a server basis.</td>
</tr>
</tbody>
</table>

Table 12-3 describes the icap service load-balancing options.

Table 12-3 icap service Load-Balancing Options

<table>
<thead>
<tr>
<th>Load-Balancing Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client IP hash</td>
<td>Uses a hash-based algorithm based on the client IP address for load balancing the ICAP servers in the cluster.</td>
</tr>
<tr>
<td>Round-robin</td>
<td>Uses the round-robin method in which ICAP servers take turns processing HTTP and FTP-over-HTTP requests.</td>
</tr>
<tr>
<td>Server IP hash</td>
<td>Uses a hash-based algorithm based on the server IP address for load balancing among the ICAP servers in the cluster.</td>
</tr>
<tr>
<td>Weighted</td>
<td>Uses a farm of ICAP servers with different load capacities.</td>
</tr>
</tbody>
</table>

Configuring an ICAP Server for Standalone Content Engines

ICAP servers process HTTP requests from clients based on the ICAP services configured at various vectoring points. ICAP servers perform content adaptation such as request or response modification and filtering of requests or responses at the configured vectoring points while processing HTTP and FTP-over-HTTP requests.

You can configure the maximum number of connections and the weight that can be handled by an ICAP server in a cluster of servers. The weight parameter represents the percentage of load that can be redirected to the ICAP server. An ICAP server with a weight of 40 denotes that this server handles 40 percent of the load. If the total weight of all ICAP servers in a load-balanced cluster exceeds 100, the percentage of load for each ICAP server is recalculated as a percentage measure represented by the weight parameters.

To configure an ICAP server for a configured ICAP service on a standalone Content Engine, use the icap service server global configuration command.

Table 12-4 describes the icap service server command parameters for configuring an ICAP server for a standalone Content Engine.
Use the `icap logging` global configuration command to configure and enable transaction logging for ICAP exchanges between the external ICAP servers and standalone Content Engines.

```
icap logging { enable | format [custom word | standard] }
```

Specify the format of the transaction log (custom or standard).
- If you want to create transaction logs in ICAP’s standard logging format, choose `standard`.
- If you want to log additional fields not included in the standard format, choose `custom`.

Note: Customized format for transaction logging is not supported in the ACNS 5.1, ACNS 5.2.x, or ACNS 5.3.x software. Only standard transaction log format is available for ICAP services configured on a Content Engine.
Displaying Information About an ICAP Configuration

To display the current ICAP configure for standalone Content Engines, use the `show icap` EXEC command. The command output shows the status of the enabled ICAP features, the service definitions, a list of vectoring points, and an ordered list of ICAP services.

To display the definition and status of a specific ICAP service that is configured on the Content Engine, use the `show icap service service-name` EXEC command.

To display an ordered list of configured ICAP services and their status, use the `show icap vector-point vector-point-name` EXEC command.

```
ContentEngine# show icap vector-point ?
    reqmod-postcache  Display reqmod-postcache information
    reqmod-precache   Display reqmod-precache information
    rspmod-precache  Display rspmod-precache information
```

Displaying Statistics for ICAP Services

To display ICAP statistics for all of the configured ICAP services, enter the `show statistics icap` EXEC command. This command has no arguments or keywords. There is no default behavior or values.
Configuring the Rules Template on Standalone Content Engines

This chapter describes how to configure the Rules Template on standalone Content Engines. The Rules Template specifies the rules by which the Content Engine filters HTTP, HTTPS, and RTSP traffic. These configured rules might rewrite certain headers, redirect the request, or otherwise manipulate the request.

This chapter contains the following sections:

- Rules Template Overview, page 13-1
- Understanding Actions and Patterns, page 13-5
- Configuring the Rules Template, page 13-22
- Displaying Statistics for Configured Rules, page 13-33
- Clearing Statistics for Configured Rules, page 13-33
- Using Regular Expressions for Rules, page 13-34

Note

The term HTTP traffic is used to refer to requests over HTTP including HTTP, FTP-over-HTTP, and HTTPS-over-HTTP. The Rules Template is not supported for FTP native requests.

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication.

Rules Template Overview

The Rules Template feature allows you to specify a set of rules, each clearly identified by a pattern and an action. This feature allows you to configure a standalone Content Engine to use specific rules to filter HTTP, HTTPS, and RTSP traffic. A common use of this feature is to configure a Content Engine to block the spread of Internet worms and viruses within an organization by checking whether a requested web page matches the pattern of a known Internet worm and if so then automatically blocking the request.

If you have enabled rules processing on a Content Engine (enabled the Rules Template feature on the Content Engine and configured rules for the Content Engine), the Content Engine checks every incoming client request to determine if a rule pattern matches the requested content. If a rule pattern matches the given request, the Content Engine uses the specified action (policy) to handle this incoming traffic.
There are two basic types of pattern comparisons: string comparisons and regular expressions. A string comparison requires an exact match. A regular expression comparison searches within a string for a substring match. The regular expression does not have to match the whole string and matching is not case sensitive. (See the “Using Regular Expressions for Rules” section on page 13-34 for more information about regular expressions.)

The Content Engine can match incoming requests against the following:

- Patterns in the IP address of the client requesting the content (source IP address), including IP address, network mask, and port list
- Patterns in the IP address of the origin web or media server (destination IP addresses), including IP address, network mask, and port list
- Regular expression of the URL
- Regular expression of the domain portion of the URL
- MIME types of the web object that the client is requesting
- Regular expressions symbolizing domain names
- Headers that are sent in the request, including:
  - “User-agent of the request,” which indicates which client software is issuing the request
  - “Referer,” which indicates the web page from which the browser jumped to this link
  - “Request Line,” which indicates the request line itself

The Rules Template feature is mostly applicable to the HTTP, FTP, and HTTPS protocols. Policies that support streaming media object protocol such as RTSP, in addition to HTTP, FTP, and HTTPS, are indicated in Table 13-1.

<table>
<thead>
<tr>
<th>Policy (Action)</th>
<th>HTTP Requests</th>
<th>WMT Requests</th>
<th>RTSP Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow the request.</td>
<td>See Table 13-2</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Append the username to request headers.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block the request.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Override the HTTP response header and cache the object.</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Cache the object depending on the HTTP response header.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bypass authentication for the request.</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Use a specific object freshness calculation factor.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset the request.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Do not cache an object.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bypass an upstream proxy for the request.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redirect the request to a different URL.</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Revalidate the object with the origin server.</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Rewrite the URL.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 13-1  Rules Template Policies (continued)

<table>
<thead>
<tr>
<th>Policy (Action)</th>
<th>HTTP Requests</th>
<th>WMT Requests</th>
<th>RTSP Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>No URL filtering for the specified HTTP and HTTPS requests.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use a specific ICAP server.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use a specific upstream proxy.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use a specific server for the request.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set ToS or DSCP in the response sent to the client.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set ToS or DSCP in the response sent to the server.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported Rule Actions per Protocol

Table 13-2 lists the rule actions per protocol that are supported by standalone Content Engines running the ACNS 5.3.1 software and later releases. An asterisk (*) indicates that a rule action is supported for that particular protocol. WCCP means transparent support. A “*1” indicates that a rule action is only supported for that particular protocol in the ACNS 5.1.5 software and later releases.

For the RTSP streaming protocol, the redirect and the redirect_url_for_cdn rule actions are supported for RTSP requests from RealMedia players but are not supported for WMT. Consequently, these two rule actions are not supported for RTSP requests from Windows Media players. For example, Windows Media Services 9 (WMS 9) supports the block, reset, rewrite, and allow rule actions for RTSP requests, but does not support the redirect and redirect_url_for_cdn rule actions for RTSP requests.
### Table 13-2  Matrix of Supported Rule Actions Per Protocol

<table>
<thead>
<tr>
<th>Rule Actions</th>
<th>HTTP-over-HTTP</th>
<th>FTP-over-HTTP</th>
<th>HTTPS-over-HTTP</th>
<th>HTTP WCCP (Native FTP)</th>
<th>FTP-WCCP (tunneled, only available ACNS 5.1.5 Software or later)</th>
<th>HTTPS-WCCP</th>
<th>RTSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>append-username-header</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>block</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cache-cookie</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cache-non-cacheable</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cache-only</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dscp</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>freshness-factor</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>insert-no-cache</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no-auth</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no-cache</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no-persistent-connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no-proxy</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no-url-filtering</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>redirect</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>redirect-url-for-cdn</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>refresh</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rewrite</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>use-dns-server</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use-icap-service</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use-proxy</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>use-server</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>use-xforward-clt-ip</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Understanding Actions and Patterns

A rule is specified by a pattern and an action.

- **A pattern** defines the limits of an incoming request; for instance, a pattern may specify that the source IP address fall in the subnet range 172.16.0.0.

  When defining a pattern, you specify the following information:
  - The pattern type (for example, “src-ip” for the source IP address)
  - The pattern value (for example, “172.16.*.*” to indicate that the source IP address must fall within this subnet range)
  - The number of the pattern list to which this pattern should be added (A pattern list is a container list of patterns that is identified by a pattern list number, for example, pattern list 10)

  For a complete list of supported patterns, see Table 13-3.

- **An action** is the policy that is applied to an incoming request if the request matches the specified pattern. An action is something that the Content Engine performs when processing a request, for instance, blocking the request, using an alternative proxy, and so forth. For a complete list of supported actions, see Table 13-4.

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 Content Engine CLI commands or the GUI. Because rules consume resources, the more rules there are defined, the more Content Engine performance may be affected. For information about the limitation on the number of rules that can be configured on a Content Engine, see the “Configuring the Rules Template” section on page 13-22.

You can use the Content Engine CLI or the GUI (as shown in Figure 13-1) to specify a pattern and the corresponding action.
Understanding Actions and Patterns

Chapter 13 Configuring the Rules Template on Standalone Content Engines

To access the Rules Template window, choose **System > Rules Templates** from the Content Engine GUI. To view detailed information about how to use this window to configure the Rules Template, click the **HELP** button.

The **rule pattern-list** global configuration command allows you to create a pattern list and add specific patterns to that pattern list. The patterns within the pattern list can be ANDed or ORed together. After defining one or more pattern lists, you can use the **rule action** global configuration command to associate specific actions with a defined pattern list. The following example shows how patterns are ANDed by configuring different patterns with the same pattern list number and applying that pattern list to an action:

```
ContentEngine(config)# rule pattern-list 1 url-regex yahoo
ContentEngine(config)# rule pattern-list 1 dst-port 80
ContentEngine(config)# rule action block pattern-list 1
```

The CLI method is used throughout the rest of this chapter to illustrate how to configure the Rules Template on a standalone Content Engine. For more information about how to use the Content Engine CLI to configure the Rules Template, see the “Configuring the Rules Template” section on page 13-22.
Supported Rule Patterns

In the ACNS 5.1 software release, three new rule patterns were added (\textit{groupname}, \textit{username}, and \textit{groupname-regex}). These new rule patterns support access control policies that are based on the group name and username of the authenticated NTLM and LDAP users. Group name-based rules apply to users who have been authenticated through NTLM and LDAP. Username-based rules apply to users who are authenticated through LDAP, NTLM, RADIUS, and TACACS+ (request authentication methods that involve a username for authentication).

The following example shows how to enable rule processing on the Content Engine (using the \texttt{rule enable} global configuration command) and then configure the Content Engine to block all end users in the Engineering group from downloading FTP URLs (FTP requests from a client browser) that contain the expression “java”:

\begin{verbatim}
ContentEngine(config)# rule enable
ContentEngine(config)# rule pattern-list 1 group-type and
ContentEngine(config)# rule pattern-list 1 groupname Engineering
ContentEngine(config)# rule pattern-list 1 url-regex java
ContentEngine(config)# rule action block pattern-list 1 protocol ftp
\end{verbatim}

\begin{itemize}
\item \textbf{Note}: Authorization through group name-based and username-based rules occurs only after HTTP request authentication and group-based access list authorization have occurred. If the configuration in the Rules Template and the access list match, the access list takes precedence.
\end{itemize}

Table 13-3 describes the types of rule patterns that you can add to a pattern list.

\begin{table}[h!]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Pattern} & \textbf{Description} \\
\hline
\texttt{domain} & Matches the domain name in the URL or the Host header against a regular expression. For example, \texttt{\.*ibm\.*} matches any domain name that contains the “ibm” substring. \texttt{\*.foo\.com\$} matches any domain name that ends with the “\.foo.com” substring. In regular expression syntax, the dollar sign “\$” metacharacter directs that a match be made only when the pattern is found at the end of a line. (See the “Using Regular Expressions for Rules” section on page 13-34 for more information about regular expressions.)

\textbf{Tip}: This rule pattern can be circumvented by users entering URLs with IP addresses instead of domain names.

\texttt{dst-ip} & Matches the request’s destination IP address and netmask against the IP address and netmask specified in the rule. For example, the following command:

\begin{verbatim}
CE(config)# rule pattern-list 2 dst-ip 10.255.0.11 255.255.255.255
\end{verbatim}

matches any transparent mode request destination IP address or the DNS-resolved IP address of the domain from the proxy mode URL. This example matches a specific host, but by modifying the mask, you can match entire subnets or groups of IP addresses.

\textbf{Tip}: To test this pattern in proxy mode requires that the Content Engine perform a DNS lookup, which can have an impact on performance.
\end{tabular}
\end{table}
Chapter 13 Configuring the Rules Template on Standalone Content Engines

Understanding Actions and Patterns

Table 13-3 Supported Types of Patterns (continued)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
</table>
| dst-port         | Matches the request’s destination port number against the port number specified in the rule. For example, the following command:  
\[CE(config)# rule pattern-list 3 dst-port 8083\]  
matches any request destined for TCP port 8083. In proxy mode, this pattern matches the port from the request URL. In transparent mode, this pattern matches the TCP destination port. |
| groupname        | Matches the group name of the end user (the web client that is requesting content), who was authenticated through LDAP or NTLM, against the group name specified in the rule. For example, specify the group name Engineering for pattern list 1, as follows:  
\[ContentEngine(config)# rule pattern-list 1 groupname Engineering\]  
This pattern can be applied only to request authentication for users who have been authenticated through LDAP or NTLM. This pattern supports exact string comparison and is case insensitive. The maximum length of the group name is 129 characters. Valid characters are an underscore and alphanumeric characters and special characters that are accepted by Active Directory. If the groupname configuration in the Rules Template and the group name-based access list match, then the access list takes precedence. |
| Tip              | If you intend to use the groupname pattern, make sure that you set the correct number of maximum group entries in the authentication group cache (the \texttt{http authentication cache max-group-entries number} global configuration command). This number should correspond to the maximum number of groups that could be returned during authorization queries (for example, the total number of groups defined on the AAA server.) The number can be from 500 to 12000. The number of entries in the authentication group cache is dependent on the physical resources available on the Content Engine. |
| groupname-regex  | Matches the group name of the end user (the web client that is requesting content) against the regular expression specified in the rule. The maximum length of the group name is 255 characters. For example, the following command:  
\[ContentEngine(config)# rule pattern-list 5 groupname-regex ING\]  
matches request-authenticated users who are members of any group that contains “ING” in its name. In this example, “ACCOUNTING” and “MARKETING” will be matched, but not “CORPORATE”. |
| group-type       | Specifies whether the pattern list is an AND or OR type. This parameter is not actually a pattern, but it defines how to handle multiple patterns in the pattern list. If the group-type is OR, then the action will be performed if any pattern in the list is matched. If the group-type is AND, then the action will be performed only if all the patterns in the list are matched. Pattern lists with multiple patterns default to the OR group-type. |
### Understanding Actions and Patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
</table>
| header-field    | Request header field pattern. Request header field patterns `referer`, `request-line`, and `user-agent` are supported for the `block`, `reset`, `redirect`, and `rewrite` actions. The `header-field referer` pattern is a regular expression comparison against the Referer HTTP header in the request. For example, the following command:
   
   ```
   CE(config)# rule pattern-list 6 header-field referer cisco
   ```
   
   matches any HTTP request that has the pattern “cisco” anywhere in the Referer header line. The `header-field request-line` pattern is a regular expression comparison against the request line of the request. For example, the following command:
   
   ```
   CE(config)# rule pattern-list 6 header-field request-line HEAD
   ```
   
   matches any request that contains the pattern “HEAD” in the request line. The `header-field user-agent` pattern is a regular expression comparison against the User-Agent header in the request. For example, the following command:
   
   ```
   CE(config)# rule pattern-list 6 header-field user-agent layer
   ```
   
   matches any request where the User-Agent header line contains the string “layer”. For instance, the rule pattern in this example will match “NSPlayer”.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
</table>
| header-field-sub | Matches the pattern in the request header and replaces the text. This pattern contains arguments for `referer`, `request-line`, and `user-agent`. The `header-field-sub referer` pattern is a regular expression comparison against the Referer HTTP header that also replaces Referer HTTP header text. For example, the following command:
   
   ```
   CE(config)# rule pattern-list 9 header-field-sub referer yahoo google
   ```
   
   matches any Referer HTTP header that contains the string “yahoo” and replaces it with “google”. The `header-field-sub request-line` pattern is a regular expression comparison against the request line that also replaces HTTP request line text. For example, the following command:
   
   ```
   CE(config)# rule pattern-list 10 header-field-sub request-line yahoo google
   ```
   
   matches any HTTP request line that contains the string “yahoo” and replaces it with “google”. The `header-field-sub user-agent` pattern is a regular expression comparison against the User-Agent header that also replaces User-Agent header text. For example, the following command:
   
   ```
   CE(config)# rule pattern-list 11 header-field-sub user-agent Mozilla NoZilla
   ```
   
   matches any request with a User-Agent header that contains the string “Mozilla” and replaces it with “NoZilla”.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icap-service-name</td>
<td>Specifies the name of the ICAP service that should be used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
</table>
| mime-type        | Matches the MIME type of the response from the origin server against the MIME type string specified in the rule. (For an explanation of mime-type, see RFC 2046 at [http://www.faqs.org/rfcs/rfc2046.html](http://www.faqs.org/rfcs/rfc2046.html).) For example, the following command:
   
   ```
   CE(config)# rule pattern-list 12 mime-type java
   ```
   
   matches any MIME type that contains the substring “java”, such as application/x-javascript.

---

Table 13-3 Supported Types of Patterns (continued)
Understanding Actions and Patterns

**src-ip**

Matches the request’s source IP address and netmask against the IP address and a netmask specified in the rule.

**Tip** If this rule is configured on an outgoing proxy, requests will be sourced from a subordinate proxy IP address. For example, the following command:

```
CE (config)# rule pattern-list 13 src-ip 10.255.11.128 255.255.255.192
```

matches requests that come from any client on the 10.255.11.128 subnet. It will not match requests from clients on the 10.255.11.64 or 10.255.11.192 subnets.

**username**

Matches the username of the end user (the web client that is requesting content), who was authenticated through LDAP, NTLM, RADIUS, or TACACS+, against the username or usernames specified in the rule. The maximum length of username is 129 characters for LDAP, RADIUS, or TACACS+ authentication. Valid characters for the username are alphanumeric characters (a-z, A-Z, 0-9) and the following special characters:

! @ # $ % ^ & ( ) - _ { } . ~ `.

However, the username cannot contain the following special characters:

\ + = : ; ? < > ,

This pattern supports exact string comparison and is case insensitive.

Specify multiple usernames in the same line for the same pattern list by using a comma-delimited string, as shown in the following example:

```
ContentEngine(config)# rule pattern-list 1 username jdoe8,dsmith7,jsmith50
```

By default, the match does not consider the domain name and matches only the username. To include the domain name as well as the username in the match, specify domainname\username, as shown in the following example:

```
ContentEngine(config)# rule pattern-list 1 username cisco\jdoe8
```

For NTLM authentication, the domain\username:password:NTLM string must be 50 characters or less. If this string is greater than 50 characters, the domain name is truncated, and the rule username pattern is not matched. An error message is generated in the system log in this situation.

To match all users in a particular domain, enter:

```
ContentEngine(config)# rule pattern-list 1 username domain domainname\*
```

*domainname* is the name of the domain (for example, cisco).
For information about how to use the `rule` global configuration command to specify patterns for a standalone Content Engine, see the “Configuring Rules for Standalone Content Engines” section on page 13-23.

**Supported Rule Actions**

To display a list of supported rule actions, enter the `rule action`? global configuration command.

For information about how to use the `rule` global configuration command to specify patterns for a standalone Content Engine, see the “Configuring Rules for Standalone Content Engines” section on page 13-23.

**Table 13-3 Supported Types of Patterns (continued)**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL-regex</td>
<td>Matches the URL against the regular expression specified in the rule. The match is case insensitive. For example, the following command:</td>
</tr>
<tr>
<td></td>
<td><code>CE(config)# rule pattern-list 14 url-regex cgi-bin</code></td>
</tr>
<tr>
<td></td>
<td>matches any URL that contains the string “cgi-bin”.</td>
</tr>
<tr>
<td></td>
<td><strong>Tip</strong> If the request was made in transparent mode, a proxy mode URL is constructed before the comparison is made.</td>
</tr>
<tr>
<td>URL-regsub</td>
<td>For the <strong>rewrite</strong> and <strong>redirect</strong> actions, matches the URL against a regular expression and forms a new URL in accordance with the pattern specified. For example, the following command:</td>
</tr>
<tr>
<td></td>
<td><code>CE(config)# rule pattern-list 15 url-regsub yahoo google</code></td>
</tr>
<tr>
<td></td>
<td>matches any URL that contains the string “yahoo” and replaces it with “google”. The match is case insensitive. The valid substitution index range is from 1 to 9.</td>
</tr>
</tbody>
</table>

For information about how to use the `rule` global configuration command to specify patterns for a standalone Content Engine, see the “Configuring Rules for Standalone Content Engines” section on page 13-23.

**Supported Rule Actions**

To display a list of supported rule actions, enter the `rule action`? global configuration command.

For information about how to use the `rule` global configuration command to specify patterns for a standalone Content Engine, see the “Configuring Rules for Standalone Content Engines” section on page 13-23.

**Table 13-4 describes the types of actions that you can associate with a defined pattern list.**
## Table 13-4  Supported Types of Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow</td>
<td>Allows incoming requests that match the pattern list. This rule action can be used in combination with reset or block actions to allow selective types of requests. Allow does not carry any meaning as a standalone action.</td>
</tr>
<tr>
<td>append-username-header</td>
<td>Appends a Unname HTTP header to the request sent to the server. This action facilitates single sign-on if the web server recognizes the Unname HTTP header. This rule action requires that RADIUS, NTLM, or LDAP request authentication be configured.</td>
</tr>
<tr>
<td>block</td>
<td>Blocks incoming requests that match the pattern list and allows all others.</td>
</tr>
<tr>
<td>cache-cookie</td>
<td>Caches requests with pattern matches that contain cookies. To be effective, this rule action also requires a corresponding rule action refresh if cookies are user-specific. Never configure this action when dynamic content is returned based on the cookie.</td>
</tr>
<tr>
<td>cache-only</td>
<td>Caches objects depending on the HTTP response headers. 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 cache-only 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 cache-only rules, the object is not cached.</td>
</tr>
<tr>
<td>cache-non-cacheable</td>
<td>Overrides the HTTP response headers and caches objects that would not normally be cached.</td>
</tr>
<tr>
<td>dscp client</td>
<td>Configures IP Type of Service/differentiated services code point (ToS/DSCP) field responses for the client. Setting the ToS or DSCP is called packet marking, allowing you to partition network data into multiple priority levels or types of service.</td>
</tr>
<tr>
<td>dscp client cache-hit</td>
<td>Cache hit responses to the DSCP client.</td>
</tr>
<tr>
<td>dscp client cache-miss</td>
<td>Cache miss responses to the DSCP client.</td>
</tr>
<tr>
<td>dscp server</td>
<td>Configures the IP ToS or DSCP code point field for requests to the origin server.</td>
</tr>
<tr>
<td>dscp server match-client</td>
<td>Uses the client’s ToS or DSCP value.</td>
</tr>
<tr>
<td>dscp server set-dscp</td>
<td>Specifies the DSCP value. For a list of DSCP values, see Table 11-1.</td>
</tr>
<tr>
<td>dscp server set-tos</td>
<td>Specifies the Type of Service (ToS) value. For a list of ToS values, see Table 11-2.</td>
</tr>
<tr>
<td>freshness-factor</td>
<td>Determines the Time To Live (TTL) if the request URL matches a specified regular expression. The refresh configuration takes priority over the freshness-factor configuration.</td>
</tr>
<tr>
<td>insert-no-cache</td>
<td>Inserts a no-cache header into the response.</td>
</tr>
<tr>
<td>no-auth</td>
<td>Does not authenticate. (The origin server might still require authentication.)</td>
</tr>
<tr>
<td>no-cache</td>
<td>Returns the requested object, but does not cache this object. If both the no-cache and selective-cache actions are matched, no-cache takes precedence.</td>
</tr>
<tr>
<td>no-persistent-connection all</td>
<td>Does not use a persistent connection for the request if the pattern is matched.</td>
</tr>
<tr>
<td>no-persistent-connection client</td>
<td>Does not use a persistent connection to client connections only.</td>
</tr>
</tbody>
</table>
### Table 13-4 Supported Types of Actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no-persistent-connection server</td>
<td>Does not use a persistent connection to server connections only.</td>
</tr>
<tr>
<td>no-proxy</td>
<td>Does not use an outgoing proxy. (This action does not preclude the request going through a transparent proxy.) For a cache miss, the server is contacted directly rather than using the configured upstream proxy. For an example of how to use this action, see the “Example of no-proxy Action” section on page 13-16.</td>
</tr>
<tr>
<td>no-url-filtering</td>
<td>Bypasses URL filtering for HTTP and HTTPS requests. This feature is supported for local list URL filtering (good and bad site lists), as well as Websense, SmartFilter, or N2H2 URL filtering in the ACNS 5.2.3 software and later releases. For an example of how to use this action, see the “Example of no-url-filtering Action” section on page 13-14.</td>
</tr>
<tr>
<td>redirect-url-for-cdn</td>
<td>Redirects the original request to an alternative URL (specified in the manifest file using the alternateURL attribute) for ACNS network content. <strong>Note</strong> This rule action is only applicable for Content Engines that are registered with a Content Distribution Manager; it is not applicable to standalone Content Engines.</td>
</tr>
<tr>
<td>redirect</td>
<td>Redirects the original request to the specified URL. The redirection is relevant to a RADIUS server only if a RADIUS server has been configured for redirect.</td>
</tr>
<tr>
<td>refresh</td>
<td>For a cache hit, forces an object freshness check with the server.</td>
</tr>
<tr>
<td>reset</td>
<td>Issues a TCP RST. This reset request is useful as a security precaution to block known virus request patterns, such as Code Red or Nimda virus requests.</td>
</tr>
<tr>
<td>rewrite</td>
<td>Rewrites the original request as a specified URL. The Content Engine searches for the rewritten URL in the 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 IP address of the new server to which the request must be sent. The original IP address derived from the WCCP redirect packet cannot be used. <strong>Note</strong> Rules that are checked after the rewrite rule will still use the original URL rather than the rewritten URL.</td>
</tr>
<tr>
<td>selective-cache</td>
<td>Caches the object if permitted by HTTP.</td>
</tr>
<tr>
<td>use-dns-server</td>
<td>Uses the specified DNS server.</td>
</tr>
<tr>
<td>use-icap-service</td>
<td>Uses the specified ICAP service for the specified pattern list.</td>
</tr>
<tr>
<td>use-proxy</td>
<td>For a cache miss, uses the specified upstream proxy. Specify the upstream proxy IP address (or domain name) and port number. For an example of how to use this action, see the “Example of use-proxy Action” section on page 13-14. <strong>Note</strong> If failover is specified and the proxy connection fails, uses the globally configured http proxy outgoing host. If failover is not specified and the proxy connection fails, no proxy is used.</td>
</tr>
<tr>
<td>use-server</td>
<td>Sends server-style HTTP requests from the Content Engine to the specified IP address and port on a cache miss. For an example of how to use this action, see the “Example of use-server Action” section on page 13-16. <strong>Note</strong> The request from the client can be a proxy style or server style request.</td>
</tr>
<tr>
<td>use-xforward-clt-ip</td>
<td>Uses the client IP address from the X-Forwarded-For HTTP header for content filtering.</td>
</tr>
</tbody>
</table>
Example of no-url-filtering Action

This example shows how to use the **no-url-filtering** action to bypass URL filtering feature with Websense URL filtering.

Specify the rule action **no-url-filtering** command and then associate it with a specific pattern list (pattern list 100). Add the domain pattern type to pattern list 100 in order to configure the Content Engine to match requests that have foo.com as the domain. In this example, Websense URL filtering has already been configured and enabled on the Content Engine:

```
ContentEngine (config)# rule action no-url-filtering pattern-list 100
ContentEngine (config)# rule pattern-list 100 domain .*foo.com
ContentEngine (config)# rule enable
```

The **no-url-filtering** action supports the following rule patterns: src-ip, dst-ip, dst-port, domain, group-name, groupname-regex, header-field, url-regex, and username. Patterns can be ANDed or ORed by using the group-type pattern (for example, **rule pattern-list 1 group-type and**). The default is OR.

When the Content Engine receives an HTTP or HTTPS request that has foo.com as the domain, the rule action **no-url-filtering** rule is matched. Consequently, the Content Engine bypasses URL filtering for that particular request as shown in this partial output of the **debug http proxy** command:

```
Oct 28 12:25:12 Content Engine 3: Rule action no-url-filtering match
  - Bypassing urlfiltering
```

If the rule action **no-url-filtering** rule is matched and SmartFilter URL filtering is being used instead of Websense URL filtering, the output of the **debug http proxy** command would be as follows:

```
Oct 28 12:25:12 Content Engine 3: Rule action no-url-filtering match
  - Bypassing SmartFilter processing
```

When the Content Engine receives an HTTP or HTTPS request for websites other than foo.com (for requests that have www.abc.com as the domain), the rule action **no-url-filtering** rule is not matched. Consequently, the Content Engine proceeds with Websense URL filtering for that particular request as shown in this partial output of the **debug http proxy** command:

```
Oct 28 12:28:06 Content Engine 3: Rule action no-url-filtering not hit - Proceed with urlfiltering
```

If the rule action **no-url-filtering** rule is not matched and SmartFilter URL filtering is being used instead of Websense URL filtering, the output of the **debug http proxy** command would be as follows:

```
Oct 28 12:25:12 Content Engine 3: Rule action no-url-filtering not hit- Proceed with SmartFilter processing
```

To display statistics for the **no-url-filtering** action, enter the **show statistics rule http action no-url-filtering** EXEC command.

To display information about the **no-url-filtering** action, enter the **show run** and **show rule all** EXEC commands.

Example of use-proxy Action

This example shows a typical use of the **use-proxy** action.

The rule action **use-proxy proxy pattern-list number** global configuration command can be used to configure only one proxy for a particular pattern list. If you attempt to configure a second proxy for the same pattern list (for example, pattern list 1), you will receive an error message that indicates the rule entry already exists.
ContentEngine12 (config)# rule action use-proxy 10.16.0.0 8080 pattern-list 1
ContentEngine12 (config)# rule action use-proxy 10.77.157.42 8080 pattern-list 1
Rule entry is duplicate
Rule use-proxy exists with IP: 10.16.0.0. Please remove it and reconfigure

When rule action use-proxy is configured with "failover" qualifier, first it would try
to connect to proxy configured in use-proxy, if it fails, it would fall back to outgoing
proxy, configured through the cli "http proxy outgoing host <host> <port>"

If the use-proxy feature is configured without failover (for example, you have entered the rule action
use-proxy 10.16.0.0 8080 pattern-list 1 command), the Content Engine will send the request to the
use-proxy (the server with the IP address of 10.16.0.0). If the Content Engine does not obtain a response
from the use-proxy, then it will send an error message to the client without failing over to the HTTP
outgoing proxy.

If the use-proxy feature is configured with failover (for example, you have entered the rule action
use-proxy 10.16.0.0 8080 failover pattern-list 1 command), the Content Engine will send the request
to the use-proxy (for example, the server with the IP address of 10.16.0.0). If the Content Engines does
not obtain a response from the use-proxy, then it fails over to the specified HTTP outgoing proxy (for
example, the server that has been specified as the primary outgoing host with the http proxy outgoing
host 10.77.157.42 8080 primary global configuration command). The following example shows a
sample configuration in which the use-proxy feature is configured with failover:

ContentEngine 12# show run
Sep  1 06:42:32 ContentEngine 12-admin-shell: %CE-PARSER-6-350232: CLI_LOG
shell_parser_log: sh run
! ACNS version 5.4.0
!
!
hostname ContentEngine 12
!
http client-no-cache-request ignore
http proxy incoming 8080
http proxy outgoing host 10.77.157.42 8080 primary
!
ftp-over-http proxy incoming 8080
!
!
< output cut >
!
rule enable
rule action use-proxy 10.16.0.0 8080 failover pattern-list 1
rule pattern-list 1 domain yahoo.com
!
<output cut >
Example of use-server Action

This example shows a typical use of the use-server action. For HTTP requests that match the specified criteria, if the Content Engine needs to contact the origin server (for example, if a cache miss occurs), the Content Engine does not go to the server indicated in the request to retrieve the requested object; instead, it uses a different destination server that is specified in the rule. This feature is primarily used for on-demand requests, and is typically used in reverse proxy deployments.

In this example, the Content Engine is a reverse proxy for www.abcbigcorp.com and is a proxy to the rest of the Internet. The IP address for the company’s website (www.abcbigcorp.com) is actually the IP address of the Content Engine, and not the company’s web site server. When the Content Engine receives the request http://www.abcbigcorp.com/main.html, its normal processing would be to obtain the IP address of www.abcbigcorp.com and send the request to that IP address. However, in this case, because the IP address of www.abcbigcorp.com is the IP address of the Content Engine, the administrator needs to prevent the Content Engine from sending the request to itself.

Consequently, the administrator of CE1 can configure the following rule that will instruct CE1 to send such requests (for example, cache misses) to the web server for www.abcbigcorp.com:

```
CE1(config)# rule use-server 1.2.3.4 80 domain www.abcbigcorp.com
```

1.2.3.4 is the IP address of the web server for www.abcbigcorp.com.

Note

This rule applies to HTTP processing only.

Example of no-proxy Action

The no-proxy action is applicable when the administrator of the Content Engine has configured an outgoing proxy server for the Content Engine. The no-proxy action states that for requests that match the criteria, if a connection with the origin server is needed (for example, because of a cache miss), the Content Engine should not use the specified proxy server to establish the connection with the origin server. This rule is useful if a company has a Content Engine (CE1) at the Internet gateway to cache all Internet content, and a Content Engine at each branch office (CE2, CE3, CE4). In this case, the administrator can configure CE2, CE3, and CE4 at the branch offices to use CE4 as their outgoing proxy server, but set up the no-proxy rule for requests for corporate internal content.

When CE2, CE3, and CE4 receive a client request and the requested content is not already stored in their local caches, they will process the request as follows:

- If the client request is for Internet content, then CE2, CE3, and CE4 should use CE1 at the Internet gateway instead of going to the origin server directly.
- If the client request is for corporate internal content, then CE2, CE3, and CE4 should establish a connection directly with the origin server instead of going to CE1.

For information about how to use the rule action global configuration command, see the “Associating an Action with an Existing Pattern List” section on page 13-30. For a list of supported action and pattern combinations, see Table 13-5 and Table 13-6. For a list of the supported rule actions per protocol, see Table 13-2.
Supported Action and Pattern Combinations

Not all actions support all patterns for request matching because some patterns do not make sense for some actions. See Table 13-5 and Table 13-6 for a list of supported action and pattern combinations.

An asterisk "*" indicates that a particular action and pattern combination is supported in the ACNS 5.2.1 software and later releases.

Table 13-5  
Supported Action and Pattern Combinations for Standalone Content Engines—Part 1

<table>
<thead>
<tr>
<th>Action</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>domain</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>allow</td>
<td>*</td>
</tr>
<tr>
<td>append-username header</td>
<td>*</td>
</tr>
<tr>
<td>block</td>
<td>*</td>
</tr>
<tr>
<td>cache-cookie</td>
<td>*</td>
</tr>
<tr>
<td>cache-non-cacheable</td>
<td>*</td>
</tr>
<tr>
<td>cache-only</td>
<td>*</td>
</tr>
<tr>
<td>dscp</td>
<td>*</td>
</tr>
<tr>
<td>dscp client</td>
<td>*</td>
</tr>
<tr>
<td>dscp server</td>
<td>*</td>
</tr>
<tr>
<td>freshness-factor</td>
<td>*</td>
</tr>
<tr>
<td>insert-no-cache</td>
<td>*</td>
</tr>
<tr>
<td>no-auth</td>
<td>*</td>
</tr>
<tr>
<td>no-cache</td>
<td>*</td>
</tr>
</tbody>
</table>

Cisco ACNS Software Configuration Guide for Locally Managed Deployments
Chapter 13 Configuring the Rules Template on Standalone Content Engines

Table 13-6 Supported Action and Pattern Combinations for Standalone Content Engines—Part 2

<table>
<thead>
<tr>
<th>Action</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>domain</td>
</tr>
<tr>
<td>no-persistent-connection</td>
<td>*</td>
</tr>
<tr>
<td>no-proxy</td>
<td>*</td>
</tr>
<tr>
<td>no-url-filtering</td>
<td>*</td>
</tr>
<tr>
<td>redirect</td>
<td>*</td>
</tr>
<tr>
<td>refresh</td>
<td>*</td>
</tr>
<tr>
<td>reset</td>
<td>*</td>
</tr>
<tr>
<td>rewrite</td>
<td>*</td>
</tr>
<tr>
<td>selective-cache</td>
<td>*</td>
</tr>
<tr>
<td>use-dns-server</td>
<td>*</td>
</tr>
<tr>
<td>use-icap-service</td>
<td>*</td>
</tr>
<tr>
<td>use-proxy</td>
<td>*</td>
</tr>
<tr>
<td>use-server</td>
<td>*</td>
</tr>
<tr>
<td>use-xforward</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>dst-ip</td>
</tr>
<tr>
<td></td>
<td>dst-port</td>
</tr>
<tr>
<td></td>
<td>header-field-referrer</td>
</tr>
<tr>
<td></td>
<td>header-field-sub-referrer</td>
</tr>
<tr>
<td></td>
<td>header-field-user-agent</td>
</tr>
<tr>
<td></td>
<td>mime-type</td>
</tr>
<tr>
<td></td>
<td>src-ip</td>
</tr>
<tr>
<td></td>
<td>url-regex</td>
</tr>
<tr>
<td></td>
<td>url-regsub</td>
</tr>
<tr>
<td></td>
<td>groupname, username,</td>
</tr>
<tr>
<td></td>
<td>groupname, regex</td>
</tr>
</tbody>
</table>

1. The CLI help does not list these supported patterns for the use-proxy rule action (CSCsb63246).

Rules Template Processing Considerations

When there are multiple rules configured on a standalone Content Engine, the rules are executed in a particular order.

To understand the processing order of rules in ACNS 5.x, you must understand the following aspects of rule processing:

- There is a predefined order of execution among the defined actions. 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. This order is predefined and is not affected by the order in which the rules were entered.

- Among rules of the same action, there is a predefined order among the rules pattern. In other words, within 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. This order is again predefined and not affected by the order in which the rules were entered.
• Among all rules of the same action, which are configured to a pattern list that contains the same type of patterns, the individual patterns are matched in the order that they were configured.

For example, if you specified the following rules, the Content Engine will first attempt to match the URLs that end with .asf:

```
ContentEngine(config)# rule action block pattern-list 1
ContentEngine(config)# rule pattern-list 1 url-regex .*\.asf$
ContentEngine(config)# rule pattern-list 1 url-regex .*\.avi$
ContentEngine(config)# rule pattern-list 1 url-regex .*\.wav$
ContentEngine(config)# rule pattern-list 1 url-regex .*\.wmv$
```

The Content Engine will then first attempt to match the URLs that end with .avi. After attempting to match the URLs that end with .avi, the Content Engine will attempt to match the URLs that end with .wav and then .wmv.

• When a new release of the ACNS software adds more rule actions and patterns (for example, the **groupon**, **username**, and **groupon-name-regex** patterns and the **cache-cookie** action were added in the ACNS 5.2.1 software), the order of processing among existing actions and patterns does not change.

For the order in which rule patterns are executed, see the “Execution Order of Rule Patterns” section on page 13-21. This order is not affected by the order in which the rules are entered using the Content Engine GUI or the CLI.

**Tip**

In the ACNS 5.x software, if you enter the *show rule* EXEC command on a standalone Content Engine, the rules are displayed randomly. However, if you enter the *show statistics rule* EXEC command, the rules are displayed in the order in which the rule actions are executed. Consequently, you can use this command to see how a standalone Content Engine will process the rules that you define for it.

### Rule Command for Converting Hostnames to IP Addresses

In the ACNS 5.3.1 software and earlier releases, the **use-proxy** rule action and the **failover** option in the **use-proxy** rule action perform hostname to IP address translation at the time of the CLI configuration. If the IP address for the specified hostname changes, the service rule would no longer function.

In the ACNS 5.3.3 software and later releases, the **rule dns-resolve each-request** global configuration command is available. When this CLI command is enabled, the caching process on the Content Engine resolves the hostname each time that it processes the request and matches the pattern for the **use-proxy** rule action and the **failover** option in the **use-proxy** rule action.

For the ACNS 5.3.3 software and later releases, the caching process uses the initially resolved IP (done at the time of the CLI configuration) for processing the **use-proxy** rule action and the **failover** option in the **use-proxy** rule action when the **rule dns-resolve each-request** CLI command is disabled. For instance, the following is an example of the CLI command syntax for the yahoo and abc websites upon configuration with the ACNS 5.3.3 software and later releases:

```
ContentEngine(config)# rule action use-proxy www.yahoo.com 8080 failover pattern-list 10
ContentEngine(config)# rule action use-proxy www.abc.com 8090 pattern-list 20
```

In contrast, the following is an example of the CLI command syntax for the yahoo and abc websites upon configuration with the ACNS 5.3.1 software and earlier releases:

```
ContentEngine(config)# rule action use-proxy 66.94.230.42 8080 failover pattern-list 10
ContentEngine(config)# rule action use-proxy 199.181.132.250 8090 pattern-list 20
```
Execution Order of Rule Actions

In the ACNS 5.2.3 software and later releases, the order in which the rule actions are executed is as follows:

1. Reset (if the pattern list contains a header-field pattern.)
2. Redirect-url-for-cdn (this action is only applicable for Content Engines that are registered with a Content Distribution Manager and is not applicable for standalone Content Engines)
3. No-auth (before authentication using RADIUS, LDAP, or NTLM)
4. Reset
5. Block / allow
6. Redirect (before cache lookup)
7. Rewrite (before cache lookup)
8. No-url-filtering
9. Refresh (after cache lookup, in the case of a cache hit)
10. Freshness-factor (after cache lookup, in the case of a cache hit)
11. Use-server
12. No-proxy
13. Use-proxy
14. Use-dns-server
15. ToS/DSCP server (ToS bits on the connection to the server)
16. ToS/DSCP client (ToS bits on the connection that the server uses to send a response to the client)
17. DSCP client cache-miss
18. DSCP client cache-hit
19. Insert-no-cache
20. No-cache
21. Cache-non-cacheable (when the response is received from the server)
22. Cache-only (when the response is received from the server)
23. Append-username-header
24. Use-icap-service
25. Use-xforward-clt-ip
26. No-persistent-connection
27. Cache-cookie
28. No-selective-cache
29. Allow

1. Allow and block carry the same precedence. The order of execution depends on the order of configuration between allow and block actions. Other actions always take precedence over allow. For example, a reset action always takes precedence over allow regardless of the order of configuration.
The reset, block, rewrite, and redirect rule actions support the following additional patterns: request-line, referer, and user-agent regular expressions. The request-line regular expression matches the first line of the request. The user-agent regular expression matches the User-Agent header value of the request. The referer regular expression matches the referer header value of the request.

In the following example, the Content Engine is configured to replace the string internal.domain.com in a request to the server named dummy:

```
ContentEngine(config)# rule rewrite header-field referer internal.domain.com dummy
```

In the following example, if an empty string is given as a replacement pattern, the referer header is stripped. This rule states that for all requests with a referer header that indicates a corporate internal server in ABCBigCorp, strip the referer field so that the outside web server will not see the name of the corporate internal server. This is a useful practice for network security.

```
ContentEngine(config)# rule rewrite header-field referer internal.abcbigcorp.com ""
```

Stripping of the referer header occurs in the user-agent pattern as well.

---

**Note**

The rule action no-proxy, rule action use-proxy hostname port-number failover, and rule action use-proxy commands take precedence over the https proxy outgoing, http proxy outgoing, and ftp proxy outgoing global configuration commands.

Among the use-server, no-proxy, and use-proxy rules, the use-server rule is the first one to be checked. If none of these rules match, the no-proxy and use-proxy rules are executed in succession (the use-proxy rule is not checked if there is a match with the no-proxy rule). 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, because 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. 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.

### Execution Order of Rule Patterns

The execution order is as follows:

1. Header-field
2. Header-field-sub
3. Other patterns: url-regsub, dst-port, src-ip, url-regex, domain, dst-ip, mime-type.

There is no execution order based on the actions with which these patterns are executed.

---

**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.

For example, in the following set of actions, the pattern-list 2 header-field pattern is executed first, and then the pattern-list 1 domain pattern. This order is followed because the rule action is taken only after the header information becomes available.

```
ContentEngine(config)# rule action block pattern-list 1
ContentEngine(config)# rule action block pattern-list 2
ContentEngine(config)# rule pattern-list 1 domain roti
ContentEngine(config)# rule pattern-list 2 header-field user-agent browser
```
In the ANDing of patterns shown in the following example, there is no execution order based on the pattern entry:

```
ContentEngine(config)# rule pattern-list 3 group-type and
ContentEngine(config)# rule action block pattern 3
ContentEngine(config)# rule pattern-list 3 dst-port 80
ContentEngine(config)# rule pattern-list 3 header-field user-agent browser
```

In the preceding example, the destination port (dst-port) is checked first and then the header field is checked.

If a particular rule has a header field with a normal type of pattern, then there is no particular execution order.

A search for a rule match with the remaining pattern list is not performed if a match has already been found. For instance, if a match for the `rule action block` action command is found with a URL-regex request, then the remaining patterns `domain`, `dst-ip`, or `MIME-type` are not searched. Not all patterns are applicable for the actions `rewrite` and `redirect`.

Rules by default are ORed together. Multiple rules may all match a request; then all actions are taken, with precedence set among conflicting actions based on the execution order of rule actions as defined in the “Execution Order of Rule Actions” section on page 13-20. A `rule action` global configuration command can contain more than one pattern as configured by the `rule pattern-list` global configuration command.

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`.

---

**Note**

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’s source IP address has been transparently replaced with the sending Content Engine’s IP address on its way 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 noncacheable, the `no-cache` rules are not checked at all, because the server already recognizes that this object is not cached. Any `no-cache` rule checks are performed only for cacheable requests.

---

## Configuring the Rules Template

When configuring rules on a standalone Content Engine, remember the following important points:

- The number of actions is unlimited.
- The maximum number of pattern lists is 512.
- The maximum number of patterns per action is 128.
- A single pattern list can up to 128 patterns of a particular pattern type.
- To enter a question mark (?) character in a rule regular expression from the Content Engine CLI, use the escape character followed by a question mark (?) character. This prevents the CLI from displaying context-sensitive help.
Chapter 13  Configuring the Rules Template on Standalone Content Engines

Configuring the Rules Template

- 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.

- Authorization through group name-based and username-based rules occurs only after HTTP request authentication and group-based access control list (ACL) authorization have occurred. If the configuration in the Rules Template and the ACL match, ACL takes precedence.

- The `rule action cache-non-cacheable` command cannot cache objects if the objects are authenticated. That is, for authenticated objects, some origin servers do not send the Last-Modified or ETag entity headers. Revalidation of those authorized objects therefore cannot be performed by the Content Engine. Those authenticated objects are served only from the origin server. If the server does send the Last-Modified and ETag headers for authenticated objects, then they are properly revalidated and served from the cache.

- The `no-auth` rules result in the display of multiple authentication windows in the following situation:
  - When the main window (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 window contains objects belonging to different domains

To avoid multiple authentication windows, enter the hidden `http avoid-multiple-auth-prompts` command in global configuration mode. Check the configuration with the hidden `show http avoid-multiple-auth-prompts` EXEC command as shown in the following example:

```
ContentEngine# show http avoid-multiple-auth-prompts
Avoiding multiple authentication prompts due to no-auth rules is enabled
```

The commands in the example are hidden because they are applicable only to this specific situation.

**Note**

You can configure a Rules Template through the CLI or Content Engine GUI (as shown in Figure 13-1).

For more information about how to configure the Content Engine for rule processing, see the following sections:

- Enabling Rules Processing, page 13-23
- Configuring Rules for Standalone Content Engines, page 13-23

### Enabling Rules Processing

By default, rules processing is disabled on a Content Engine. To enable rules processing on a standalone Content Engine, use the `rule` global configuration command, as follows:

```
ContentEngine(config)# rule enable
```

### Configuring Rules for Standalone Content Engines

To set the rules by which the standalone Content Engine filters HTTP, HTTPS, and RTSP traffic, use the `rule` global configuration command:

```
rule { action action-type pattern-list list_num [protocol { all | protocol-type } ] | enable | pattern-list list_num pattern-type }
```
Table 13-7 describes the parameters for the `rule` command. If the rule command parameter is an action type or a pattern type, it is noted in Table 13-7.

**Table 13-7 Parameters for the rule CLI Command**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Action or Pattern Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td></td>
<td>Configures the action that the rule is to take if an incoming request matches the specified pattern.</td>
</tr>
<tr>
<td>action-type</td>
<td></td>
<td>Specifies the type of action (for example, <code>allow</code>) that is to be performed on an incoming request if the request matches the specified pattern.</td>
</tr>
<tr>
<td>allow</td>
<td>Action type</td>
<td>Allows the request.</td>
</tr>
<tr>
<td>pattern-list</td>
<td></td>
<td>Configures a pattern list.</td>
</tr>
<tr>
<td>list_num</td>
<td></td>
<td>Pattern list number (1–512).</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol</td>
<td>Protocol for which this rule is to be matched.</td>
</tr>
<tr>
<td>all</td>
<td></td>
<td>Matches this rule with all applicable protocols for this action.</td>
</tr>
<tr>
<td>protocol-type</td>
<td>Protocol type</td>
<td>Specifies the protocol type (type of incoming traffic) for which this rule is to be matched.</td>
</tr>
<tr>
<td>enable</td>
<td></td>
<td>Enables rules processing.</td>
</tr>
<tr>
<td>http</td>
<td>Protocol</td>
<td>Matches incoming HTTP traffic against this rule.</td>
</tr>
<tr>
<td>https</td>
<td>Protocol</td>
<td>Matches this rule for incoming HTTPS traffic.</td>
</tr>
<tr>
<td>rtp</td>
<td>Protocol</td>
<td>Matches incoming RTSP traffic against this rule.</td>
</tr>
<tr>
<td>append-username-header</td>
<td>Action type</td>
<td>Appends the username to the request headers.</td>
</tr>
<tr>
<td>block</td>
<td>Action type</td>
<td>Blocks the request.</td>
</tr>
<tr>
<td>cache-cookie</td>
<td>Action type</td>
<td>Caches requests that contain cookies.</td>
</tr>
<tr>
<td>groupname</td>
<td>Match string with the group name (for example, Engineering). This groupname-based rules policy can be applied only to request authentication for users who are authenticated through LDAP or NTLM.</td>
<td></td>
</tr>
<tr>
<td>groupname</td>
<td>String of group name.</td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>Matches string against specified username. This username-based rules policy can be applied to any of the supported request authentication method that involves a username for authentication (for example, LDAP, NTLM, RADIUS, and TACACS+).</td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>Username string (for example, jdoe8).</td>
<td></td>
</tr>
<tr>
<td>groupname-regex</td>
<td>Match regular expression with the group name.</td>
<td></td>
</tr>
</tbody>
</table>

**Note** Most actions do not have any parameters. Exceptions to this are the `use-server`, `freshness-factor`, and `use-proxy` actions.
### Table 13-7 Parameters for the rule CLI Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Action or Pattern Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupname-regex</td>
<td></td>
<td>Regular expression to be matched with the group name.</td>
</tr>
<tr>
<td>cache-non-cacheable</td>
<td>Action type</td>
<td>Overrides HTTP response headers and caches this object.</td>
</tr>
<tr>
<td>ttl</td>
<td></td>
<td>Time To Live value of this object.</td>
</tr>
<tr>
<td>days</td>
<td></td>
<td>Time To Live units in days.</td>
</tr>
<tr>
<td>days</td>
<td></td>
<td>Time To Live value in days (1–1825).</td>
</tr>
<tr>
<td>hours</td>
<td></td>
<td>Time To Live units in hours.</td>
</tr>
<tr>
<td>hours</td>
<td></td>
<td>Time To Live value in hours (1–43800).</td>
</tr>
<tr>
<td>minutes</td>
<td></td>
<td>Time To Live units in minutes.</td>
</tr>
<tr>
<td>minutes</td>
<td></td>
<td>Time To Live value in minutes (1–2628000).</td>
</tr>
<tr>
<td>seconds</td>
<td></td>
<td>Time To Live units in seconds.</td>
</tr>
<tr>
<td>seconds</td>
<td></td>
<td>Time To Live value in seconds (1–157680000).</td>
</tr>
<tr>
<td>cache-only</td>
<td>Action type</td>
<td>Caches this object only.</td>
</tr>
<tr>
<td>dscp client</td>
<td>Action type</td>
<td>Configures IP Type of Service or differentiated services code point (ToS/DSCP) field responses for the client.</td>
</tr>
<tr>
<td>cache-hit</td>
<td></td>
<td>Sends responses to the client when a cache hit occurs.</td>
</tr>
<tr>
<td>match-server</td>
<td></td>
<td>Uses the original ToS or DSCP value of the server.</td>
</tr>
<tr>
<td>set-dscp</td>
<td></td>
<td>Configures differentiated services code point (DSCP) values. For a list of DSCP values, see Table 11-1.</td>
</tr>
<tr>
<td>set-tos</td>
<td></td>
<td>Configures Type of Service (ToS) values. For a list of ToS values, see Table 11-2.</td>
</tr>
<tr>
<td>cache-miss</td>
<td></td>
<td>Sends responses to the client when a cache miss occurs.</td>
</tr>
<tr>
<td>dscp server</td>
<td>Action type</td>
<td>Configures the ToS or DSCP services code point for outgoing responses.</td>
</tr>
<tr>
<td>match-client</td>
<td></td>
<td>Uses the original ToS or DSCP value of the client.</td>
</tr>
<tr>
<td>enable</td>
<td></td>
<td>Enables rules processing.</td>
</tr>
<tr>
<td>freshness-factor</td>
<td>Action type</td>
<td>Caches heuristic modifiers.</td>
</tr>
<tr>
<td>exp_time</td>
<td></td>
<td>Expiration time of object as a percentage of age (0–100).</td>
</tr>
<tr>
<td>insert-no-cache</td>
<td>Action type</td>
<td>Inserts a no-cache header in the response.</td>
</tr>
<tr>
<td>no-auth</td>
<td></td>
<td>Does not authenticate.</td>
</tr>
<tr>
<td>no-cache</td>
<td>Action type</td>
<td>Does not cache the object.</td>
</tr>
<tr>
<td>no-persistent connection</td>
<td>Action type</td>
<td>Prevents the use of persistent connections.</td>
</tr>
</tbody>
</table>
### Table 13-7  Parameters for the rule CLI Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Action or Pattern Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Action type</td>
<td>Prevents the use of persistent connections to either clients or servers.</td>
</tr>
<tr>
<td>client-only</td>
<td>Action type</td>
<td>Prevents the use of persistent connections to clients.</td>
</tr>
<tr>
<td>server-only</td>
<td>Action type</td>
<td>Prevents the use of persistent connections to servers.</td>
</tr>
<tr>
<td>no-proxy</td>
<td>Action type</td>
<td>Does not use any upstream proxy. For an example of how to use this rule action, see the “Example of no-proxy Action” section on page 13-16.</td>
</tr>
<tr>
<td>no-url-filtering</td>
<td>Action type</td>
<td>Bypasses URL filtering for certain HTTP and HTTPS requests. This feature is supported for local list URL filtering (good and bad site lists), as well as Websense, SmartFilter, or N2H2 URL filtering in the ACNS 5.2.3 software and later releases. For an example of how to use this rule action, see the “Example of no-url-filtering Action” section on page 13-14.</td>
</tr>
<tr>
<td>redirect</td>
<td>Action type</td>
<td>Redirects the request to a rewritten URL.</td>
</tr>
<tr>
<td>url</td>
<td>Action type</td>
<td>Redirect URL.</td>
</tr>
<tr>
<td>redirect-url-for-cdn</td>
<td>Action type</td>
<td>Redirects the request to an alternative URL for ACNS network content. This is applicable only for Content Engines that are registered with a Content Distribution Manager; it is not applicable for standalone Content Engines.</td>
</tr>
<tr>
<td>refresh</td>
<td>Action type</td>
<td>Revalidates the object with the web server.</td>
</tr>
<tr>
<td>reset</td>
<td>Action type</td>
<td>Issues a TCP RST.</td>
</tr>
<tr>
<td>rewrite</td>
<td>Action type</td>
<td>Rewrites the original request as a specified URL and fetches the rewritten URL on a cache miss.</td>
</tr>
<tr>
<td>selective-cache</td>
<td>Action type</td>
<td>Caches this object if permitted by HTTP.</td>
</tr>
<tr>
<td>use-dns-server</td>
<td>Action type</td>
<td>Uses a specific DNS server.</td>
</tr>
<tr>
<td>hostname</td>
<td>Pattern type</td>
<td>Hostname of the DNS server.</td>
</tr>
<tr>
<td>ip-address</td>
<td>Pattern type</td>
<td>IP address of the DNS server.</td>
</tr>
<tr>
<td>use-icap-service</td>
<td>Action type</td>
<td>Uses a specific ICAP server.</td>
</tr>
<tr>
<td>icap-service-name</td>
<td>Pattern type</td>
<td>Uses ICAP service name.</td>
</tr>
<tr>
<td>service name</td>
<td>Pattern type</td>
<td>Name of the ICAP service.</td>
</tr>
<tr>
<td>use-proxy</td>
<td>Action type</td>
<td>Makes use of a specific upstream proxy. For an example of how to use this rule action, see the “Example of use-proxy Action” section on page 13-14.</td>
</tr>
<tr>
<td>hostname</td>
<td>Pattern type</td>
<td>Hostname of the specific proxy.</td>
</tr>
<tr>
<td>ip-address</td>
<td>Pattern type</td>
<td>IP address of the specific proxy.</td>
</tr>
<tr>
<td>port</td>
<td>Pattern type</td>
<td>Port number of the specific proxy (1–65535).</td>
</tr>
<tr>
<td>use-server</td>
<td>Action type</td>
<td>Makes use of a specific server. For an example of how to use this rule action, see the “Example of use-server Action” section on page 13-16.</td>
</tr>
</tbody>
</table>
Table 13-7  Parameters for the rule CLI Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Action or Pattern Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use-xforward-clt-ip</td>
<td></td>
<td>Uses the client IP address in the x-forwarded header for filtering.</td>
</tr>
<tr>
<td>hostname</td>
<td></td>
<td>Hostname of the specific server.</td>
</tr>
<tr>
<td>ip-address</td>
<td></td>
<td>IP address of the specific server.</td>
</tr>
<tr>
<td>port</td>
<td></td>
<td>Port number of the specific server (1–65535).</td>
</tr>
<tr>
<td>domain</td>
<td>Pattern type</td>
<td>Regular expression to match the domain name.</td>
</tr>
<tr>
<td>dn_regexp</td>
<td></td>
<td>Regular expression to be matched with the domain name.</td>
</tr>
<tr>
<td>dst-ip</td>
<td>Pattern type</td>
<td>Destination IP address of the request.</td>
</tr>
<tr>
<td>d_ipaddress</td>
<td></td>
<td>Destination IP address of the request.</td>
</tr>
<tr>
<td>d_subnet</td>
<td></td>
<td>Destination IP subnet mask.</td>
</tr>
<tr>
<td>dst-port</td>
<td>Pattern type</td>
<td>Destination port number.</td>
</tr>
<tr>
<td>port</td>
<td></td>
<td>Destination port number (1–65535).</td>
</tr>
<tr>
<td>group-type</td>
<td>Pattern type</td>
<td>Specifies whether the pattern list is an AND or OR type. The default is OR.</td>
</tr>
<tr>
<td>and</td>
<td></td>
<td>Specifies an AND pattern for the pattern list.</td>
</tr>
<tr>
<td>or</td>
<td></td>
<td>Specifies an OR pattern for the pattern list.</td>
</tr>
<tr>
<td>header-field</td>
<td>Pattern type</td>
<td>Request header field pattern.</td>
</tr>
<tr>
<td>referer</td>
<td></td>
<td>Referrer request header.</td>
</tr>
<tr>
<td>ref_regexp</td>
<td></td>
<td>Regular expression to be matched with the referer request header.</td>
</tr>
<tr>
<td>request-line</td>
<td></td>
<td>Request method line.</td>
</tr>
<tr>
<td>req_regexp</td>
<td></td>
<td>Regular expression to be matched with the request method line.</td>
</tr>
<tr>
<td>user-agent</td>
<td></td>
<td>User agent request header.</td>
</tr>
<tr>
<td>ua_regexp</td>
<td></td>
<td>Regular expression to be matched with the User Agent request header.</td>
</tr>
<tr>
<td>header-field-sub</td>
<td>Pattern type</td>
<td>Requests header field pattern and substitutes replacement pattern.</td>
</tr>
<tr>
<td>referer</td>
<td></td>
<td>Referrer request header.</td>
</tr>
<tr>
<td>ref_regexp</td>
<td></td>
<td>Regular expression to be matched with the referer request header.</td>
</tr>
<tr>
<td>ref_sub</td>
<td></td>
<td>Request header regular expression replacement string.</td>
</tr>
<tr>
<td>request-line</td>
<td></td>
<td>Request method line.</td>
</tr>
<tr>
<td>req_regexp</td>
<td></td>
<td>Regular expression to be matched with the request method line.</td>
</tr>
<tr>
<td>req_sub</td>
<td></td>
<td>Request method line regular expression replacement string.</td>
</tr>
<tr>
<td>user-agent</td>
<td></td>
<td>User Agent request header.</td>
</tr>
<tr>
<td>ua_regexp</td>
<td></td>
<td>Regular expression to be matched with the User Agent request header.</td>
</tr>
</tbody>
</table>
Configuring the Rules Template

Chapter 13  Configuring the Rules Template on Standalone Content Engines

Configuring the Rules Template

With the ACNS 5.x software, 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 a cache hit
- Responses to the client on a cache miss

The ToS or DSCP may be set based on any of the policies matching the src-ip s_ipaddress s_subnet, dst-ip d_ipaddress d_subnet, dst-port port, domain LINE, url-regex LINE, or mime-type LINE options. You can also now configure global ToS or DSCP settings with the ip dscp command.

Configuring a Pattern List

To create a pattern list on a standalone Content Engine, use the rule pattern-list global configuration command as follows:

```
ContentEngine(config)# rule pattern-list list_num
```

where:

- `list_num` is the pattern list number (1-512).

For example, create pattern list 10 as follows:

```
ContentEngine(config)# rule pattern-list 10
```
Chapter 13  Configuring the Rules Template on Standalone Content Engines

Configuring the Rules Template

Adding a Pattern to an Existing Pattern List

To add a new pattern to an existing pattern list on a standalone Content Engine, follow these steps:

**Step 1** Add a pattern to an existing pattern list.

```
ContentEngine(config)# rule pattern-list list_num pattern type pattern value
```

The following example shows how to add a pattern to pattern list 10. Using the `dst-ip` (destination IP address) pattern type, this pattern will perform an action yet to be defined on the destination IP address 172.16.25.25.

```
ContentEngine(config)# rule pattern-list 10 dst-ip 172.16.25.25 255.255.255.0
```

**Note** For a complete list of supported pattern types, see Table 13-3.

**Step 2** Verify that the new pattern has been added to the specified pattern list.

The following example shows how to verify that the pattern created in **Step 1** has been added to pattern list 10:

```
ContentEngine# show rule pattern-list 10 all
Rules Template Configuration
-----------------------------
Rule Processing Enabled
-----------------------------
Pattern-Lists :
  rule pattern-list 10 dst-ip 172.16.25.25 255.255.255.0
```

For information about how to associate an action with a pattern list, see the next section, “Associating an Action with an Existing Pattern List.”
Associating an Action with an Existing Pattern List

To associate an action with an existing pattern list on a standalone Content Engine, follow these steps:

**Step 1**  
Associate an action with an existing pattern list.

```
ContentEngine(config)# rule action action-type pattern-list list_num
protocol {protocol-type | all}
```

Actions can be applied to specific protocols or to a set of protocols. If no protocol is configured, then the specified action will be taken for all the traffic that goes through the Content Engine. In the following example, the `block` action is associated with pattern list 10 for all protocols:

```
ContentEngine(config)# rule action block pattern-list 10 protocol all
```

**Note**  
Most actions do not have any parameters, as is the case with the `block` action. For more information about the parameters of the `rule action` global configuration command, see Table 13-7.

**Step 2**  
Verify that the new action has been associated with the specified pattern list.

```
ContentEngine# show rule action action-type protocol {protocol-type | all}
```

The following example shows how to verify that the `block` action has been associated with pattern list 10.

```
ContentEngine# show rule action block
Rules Template Configuration
-----------------------------
Rule Processing Enabled
Actions :
rule action block pattern-list 10 protocol all
ContentEngine#
```

Verifying an Action Performed on a Pattern List

To confirm that a certain action is performed on the specified pattern list, display the local Rules Template configuration statistics after an action has been performed.

```
ContentEngine# show statistics rule action action-type
```

In the following example, the `rule action block` command is configured and associated with an existing pattern list, which lists as its pattern the domain named yahoo.com:

```
ContentEngine(config)# rule action block pattern-list 10 protocol all
ContentEngine# show statistics rule action block
Rules Template Statistics
-------------------------
Rule hit count = 3 Rule: rule action block pattern-list 10 protocol all
ContentEngine#
```

In this example, the statistics (Rule hit count) indicate that the request to yahoo.com was denied three times.
Examples of Configuring Rules for Standalone Content Engines

This section provides a set of configuration examples for configuring rules on standalone Content Engines.

Note

In the following examples, it is assumed that all actions and patterns apply to all protocols unless specifically stated.

- To specify domains that contain .foo.com, use the domain pattern type and add this pattern to pattern list 12.

```
ContentEngine(config)# rule pattern-list 12 domain \.foo\.com
```

Associate the block action with pattern list 12 to configure the Content Engine to block all URL requests to domains that contain .foo.com.

```
ContentEngine(config)# rule action block pattern-list 12
```

Configure multiple patterns in the same pattern list (pattern list 12). If any of them matches the incoming request, the corresponding action is taken. In the following example, the rule action block global configuration command (action) blocks all patterns that are specified in pattern list 12:

```
ContentEngine(config)# rule pattern-list 12 domain \.foo\.com
ContentEngine(config)# rule pattern-list 12 dst-ip 172.16.25.25 255.255.255.0
ContentEngine(config)# rule action block pattern-list 12
```

- Configure the Content Engine not to cache any URL request that contains the *cgi-bin* string.

```
ContentEngine(config)# rule pattern-list 13 url-regex \.*cgi-bin\.*
ContentEngine(config)# rule action no-cache pattern-list 13
```

- Use no in front of a rule action global configuration command to delete rules.

```
ContentEngine(config)# no rule use-proxy foo.com 8080 pattern-list 13
ContentEngine(config)# no rule action block pattern-list 2
```

- Set the Content Engine freshness factor for MIME-type images.

```
ContentEngine(config)# rule pattern-list 13 mime-type image/.*
ContentEngine(config)# rule action freshness-factor 75 pattern-list 13
```

- Set the ToS value on the Content Engine to minimum delay for outbound requests to the destination IP address 10.1.1.1.

```
ContentEngine(config)# rule action dscp server set-tos min-delay protocol all
ContentEngine(config)# rule pattern-list 2 dst-ip 10.1.1.1 255.255.255.255
```

- Set the ToS value on the Content Engine to minimum delay for all outbound requests.

```
ContentEngine(config)# ip dscp server set-tos min-delay
```

- Configure the Content Engine to use the ToS or DSCP value that was originally sent by the server (when the object was first fetched) for all future cache hit responses for the same object.

```
ContentEngine(config)# ip dscp client cache-hit match-server
ContentEngine(config)# rule action no-cache pattern-list 3 protocol all
ContentEngine(config)# rule pattern-list 3 url-regex \.*cgi-bin\.*
ContentEngine(config)# rule pattern-list 4 dst-ip 172.31.120.0 255.255.192.0
```
- Configure the Content Engine to redirect requests for old-domain-name that has been changed to new-domain-name to the new domain name.

```
ContentEngine(config)# rule action redirect http://old-domain-name/ pattern-list 1 protocol http
ContentEngine(config)# rule pattern-list 1 url-regsub http://old-domain-name/http://new-domain-name/
```

- Configure the Content Engine to redirect requests from an IETF site to one that is locally mirrored.

```
ContentEngine(config)# rule action redirect http://www.ietf.org/rfc/(.*) pattern-list 2 protocol http
```

In the following 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.

```
```

- Configure the Content Engine to redirect all requests for linux.org to a local server in India that is closer to where the Content Engine is located.

```
ContentEngine(config)# rule action redirect http://linux.org/(.*) pattern-list 3 protocol http
```

- The rule action no-auth global configuration command permits specific login and content requests to bypass authentication and authorization features such as LDAP, RADIUS, SSH, or TACACS+. In the following example, any requests from the source IP address (src-ip) 172.16.53.88 are not authenticated:

```
ContentEngine(config)# rule enable
ContentEngine(config)# rule action no-auth pattern-list 1 protocol all
ContentEngine(config)# rule pattern-list 1 src-ip 172.16.53.88 255.255.255.255
```

- If the ACNS 5.x software is configured for authentication and SmartFilter URL filtering, requests that are allowed to bypass authentication will also bypass the SmartFilter URL filter. Configure the Content Engine not to authenticate any requests to the destination IP address (dst-ip) of 172.22.73.34.

```
ContentEngine(config)# rule action no-auth pattern-list 2 protocol all
ContentEngine(config)# rule pattern-list 2 dst-ip 172.22.73.34 255.255.255.255
```

- Configure the Content Engine not to authenticate any requests with the destination port (dst-port) of 9090.

```
ContentEngine(config)# rule action no-auth pattern-list 3 protocol all
ContentEngine(config)# rule pattern-list 3 dst-port 9090
```

- Configure the Content Engine not to authenticate any requests with cgi-bin in the URL.

```
ContentEngine(config)# rule action no-auth pattern-list 4 protocol all
ContentEngine(config)# rule pattern-list 4 url-regex .*cgi-bin.*
```

- Configure the Content Engine not to authenticate any requests that have cisco.com as the domain. For example, requests for roti.cisco.com or badal.cisco.com are excluded from the Content Engine authentication.

```
ContentEngine(config)# rule action no-auth pattern-list 5 protocol all
ContentEngine(config)# rule pattern-list 5 domain cisco.com
```
Displaying Statistics for Configured Rules

To display statistics for the rules that are configured on a standalone Content Engine, use the `show statistics rule` EXEC commands. This command lists the rule actions in their execution order. The output also shows the number of times each action was applied (rule hit counts).

In the ACNS 5.3.1 software release, the options of the `show statistics rule` command were changed to enable you to display statistics for RTSP and WMT RTSP rules.

In the ACNS 5.2.x software and earlier releases, the command options for the `show statistics rule` command were as follows:

```
ContentEngine# show statistics rule ?
   all   Display statistics of all the Rules
   http  Display statistics of http/https/all Rules
   wmt   Display statistics of wmt Rules
```

In ACNS 5.5.1 software and later releases, the command options for the `show statistics rule` command are as follows:

```
ContentEngine# show statistics rule ?
   all           Display statistics of all the Rules
   http          Display statistics of http/https/wmt-http Rule Actions
   pattern-list  Display statistics of Rule Pattern lists
   rtsp          Display statistics of rtsp/wmt-rtsp Rule Actions
```

For example, enter the `show statistics rule rtsp` command to display statistics for RTSP rules (rules for RTSP requests from RealMedia players [the RTSP rules] and rules for RTSP requests from Windows Media 9 players [the WMT-RTSP rules]).

To display the total number of rule hit counts, enter the `show statistics rule all` EXEC command. To display statistics for a specific configured rule, enter the `show statistics rule http action rule action name`. For example, to display statistics for the `no-url-filtering` rule action, enter the `show statistics rule http action no-url-filtering` EXEC command.

**Tip**

A hit counter will not increase if there is no reason to try the rule. For example, if a request is already cacheable, then the cache-non-cacheable action will never be tested.

Clearing Statistics for Configured Rules

To clear the statistics for the rules that are configured on a standalone Content Engine, use the `clear statistics rule` EXEC commands. In the ACNS 5.5.1 software and later releases, the command options for the `clear statistics rule` command are as follows:

```
ContentEngine# clear statistics rule ?
   action   Clear statistics of all the rules with same action
   all      Clear statistics of all the rules
   pattern  Clear statistics of Pattern lists
   rtsp     Clear statistics of rtsp/wmt-rtsp rules
```

For example, enter the `clear statistics rule rtsp` command to clear the statistics for the configured RTSP rules (rules configured for RTSP requests from RealMedia players [the RTSP rules] and rules configured for RTSP requests from Windows Media 9 players [the WMT-RTSP rules]).
Using Regular Expressions for Rules

The rules facility uses the Regex libraries to perform regular expression matching. A regular expression does not have to match the whole string, but searches within the string for a substring match. Matching for regular expressions is not case sensitive.

Special Characters in Regular Expressions

Whereas ordinary characters are those that match themselves in a regular expression, special characters are those used to match other characters or that have special meaning. Table 13-8 describes the special characters used in regular expressions for rules processing.

Table 13-8 Special Characters

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Period</td>
<td>Matches any character except a newline or null character.</td>
</tr>
<tr>
<td>^</td>
<td>Caret</td>
<td>Matches the beginning of the line.</td>
</tr>
<tr>
<td>$</td>
<td>Dollar sign</td>
<td>Matches the end of a string.</td>
</tr>
</tbody>
</table>

Repetition Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Asterisk</td>
<td>Matches zero (0) or more of the preceding character (greedy).</td>
</tr>
<tr>
<td>+</td>
<td>Plus sign</td>
<td>Matches one (1) or more of the preceding character (greedy).</td>
</tr>
<tr>
<td>?</td>
<td>Question mark</td>
<td>Matches zero (0) or one (1) of the preceding character.</td>
</tr>
<tr>
<td>{}</td>
<td>Braces</td>
<td>Interval operator matches specified number of occurrences of the preceding character: {count}, {min}, {min, max}</td>
</tr>
</tbody>
</table>

Listing and Grouping Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<pre><code>   | Alternation. Matches union of expressions. |
</code></pre>
<p>| []      | Brackets                     | List of single elements. Matches any element form the list. Within a bracketed list, a hyphen (-) is a range operator, a caret (^) at the beginning of a list makes a non-matching list, and other characters within a list are ordinary. |
| ()      | Parenthesis                  | Used for grouping and provides indexing.                                 |
| #      | Backslash and digit          | Indicates the index (1-9) based on the order of the groupings. Note: The grouping must be part of the match for the index to be valid. If the grouping is not part of the match, the the expression that uses the index will be invalid. |</p>

Table 13-9 Special Characters Exclusive to ACNS Software

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Exclamation point</td>
<td>Used to NOT a string. Note: The ACNS software recognizes the exclamation point (!) although it is not a regular expression operator.</td>
</tr>
</tbody>
</table>
Rules Caveats and Performance Guidelines

This section describes several group-type caveats and provides configuration guidelines for best performance.

Pattern List Group-Type Caveats

When configuring a rule pattern list, consider the following usage restrictions:

- A rule pattern list can be only one group type. It can be an AND group type, or it can be an OR group type; it cannot be both.
- You cannot AND a mime-type pattern with other pattern types.
- You cannot OR between groupname, groupname-regex, and username patterns.
- You cannot OR between substitution patterns and non-substitution patterns.
- You cannot OR between multiple substitution patterns.

Rules Configuration Guidelines for Best Performance

Rules impact performance and CPU utilization. Consider the following guidelines for best performance when configuring rules:

- Keep it simple.
  - Remember that rules are used as the exception to override default functionality.
  - Lengthy and complicated rules often lead to problems.
  - A rule list that is too long quickly becomes difficult to manage and troubleshoot.
- Reduce, eliminate, or combine rule patterns where possible.
  - Some rule patterns can be combined. For example, you can list multiple domains in the domain pattern by using the bar symbol (|) as a delimiter.
  - Keep the number of pattern-lists below the maximum: 512.
  - Keep the number of patterns per pattern-list below the maximum: 128.
- Order rules so that the most commonly applied rules are tested first.
  - You can use the `show statistics rule` EXEC command to view the rule hit count and learn how often rule actions are tested.
  - When ordering rules, keep in mind that rules of the same action, and rule patterns in the same pattern-list generally execute in the order in which they were configured.
- Regular expression rules have the greatest impact on performance.
  - Avoid the use of wild cards. Regular expression wild cards are greedy; they attempt to match 0 to 1 or more of the preceding character. Because “or more” is stated, the rule tries to match as large a string as possible.
  - Do not use wild cards at the beginning or end of an expression. You want to match a pattern within a string rather than match the entire string.
- The dst-ip pattern in a proxy mode environment requires DNS lookups that impact performance.
Chapter 13  Configuring the Rules Template on Standalone Content Engines

Rules Caveats and Performance Guidelines
PART 4

Advanced Configuration of Standalone Content Engines
Configuring Primary and Backup Proxy Servers for Standalone Content Engines

This chapter describes how to configure primary and backup (failover) proxy servers for standalone Content Engines that are running the ACNS 5.4.1 software and later releases.

This chapter contains the following sections:

- Configuring Primary Proxy Failover, page 14-2
- Designating a Primary Outgoing HTTP Proxy Server, page 14-3
- Designating a Primary Outgoing FTP Proxy Server, page 14-3
- Designating a Primary Outgoing HTTPS Proxy Server, page 14-4
- Configuring HTTP and HTTPS Outgoing Proxy Exclusion Settings, page 14-5
- Monitoring Outgoing Proxy Servers and Statistics, page 14-7
- Displaying the Current Outgoing Proxy Server Configuration, page 14-8
- Displaying Outgoing Proxy Server Statistics, page 14-8

**Note**

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication.

For information about configuring primary and backup proxy servers for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Configuring Primary Proxy Failover

For HTTP proxy caching, there is a primary proxy failover option that you can configure on standalone Content Engines. This feature is referred to as the HTTP proxy failover feature. With this feature, you can configure the forward proxy server to contact up to eight other proxy servers (outgoing proxy servers) when an HTTP cache miss occurs (that is, when the requested HTTP content is not already stored locally in the Content Engine cache).

You can use the `http proxy outgoing` global configuration command to configure up to eight backup Content Engines or any standard proxy servers for the HTTP proxy failover feature. These outgoing proxy servers can be other Content Engines or standard proxy servers that can be contacted to process HTTP cache misses without using ICP or WCCP. The function of these outgoing proxy servers is to process the HTTP cache misses that have been forwarded to them by the forwarding proxy server. 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.

Failover occurs in the order that 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 if you have used the `http proxy outgoing origin-server` global configuration 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.

Note

At any one time, the Content Engine uses only one of the configured outgoing proxy servers. They cannot be used simultaneously. The state of the outgoing HTTP proxy servers can be viewed in syslog NOTICE messages and with the `show http proxy` EXEC command.

By default, the Content Engine strips the hop-to-hop 407 (Proxy Authentication Required) error code sent by the Internet proxy. If you enter the `http proxy outgoing preserve-407` global configuration command on a standalone Content Engine, the Content Engine sends the 407 error code to the requesting client browser, and the Internet proxy authenticates the client.

Requests with a destination specified in the `proxy-protocols outgoing-proxy exclude` global configuration command bypass the primary outgoing proxy server and the failover proxy servers.

If all of the outgoing proxy servers fail to process the HTTP cache miss, the following occurs:

- If the `http proxy outgoing origin-server` option is enabled, then the Content Engine (forward proxy server) forwards the HTTP cache miss request to the origin server that was specified in the original HTTP request from the client browser.
- If the `http proxy outgoing origin-server` option is not enabled, an error is sent to the requesting client browser. Response errors and read errors are returned to the requesting client browser, because it is not possible to detect whether these errors are generated at the origin server or at the proxy server.

Note

In the ACNS 5.1 software and earlier releases, the primary proxy failover feature supported HTTP only, not HTTPS or FTP. In the ACNS 5.2.1 software and later releases, the primary proxy failover feature is supported for HTTPS-over-HTTP and FTP-over-HTTP.
The `no http proxy outgoing connection-timeout` option causes the timeout to be set to the default value of 300 milliseconds.

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

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

Requests with a destination specified in the `proxy-protocols outgoing-proxy exclude` global configuration command bypass the primary outgoing proxy and the failover proxy servers.

## Designating a Primary Outgoing HTTP Proxy Server

To configure a standalone Content Engine to direct all HTTP miss traffic to a parent cache without using the Internet Cache Protocol (ICP) or WCCP, you must explicitly designate a proxy server as the primary outgoing HTTP proxy server for the Content Engine.

To designate a proxy server as the primary outgoing HTTP proxy server for the Content Engine, use the `http proxy outgoing host host port primary` global configuration command:

- `host` is the hostname or IP address of the outgoing HTTP proxy server.
- `port` is the port number designated by the outgoing (upstream) HTTP server to accept proxy requests.

To set the specified host as the primary outgoing HTTP proxy server, use the `primary` keyword. If several servers (hosts) are configured with the `primary` keyword, the last one configured becomes the primary outgoing HTTP proxy server for the Content Engine.

In this example, host 10.1.1.1 on port 8088 is explicitly designated as the primary outgoing HTTP proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing HTTP proxy server.

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

In the ACNS 5.2.x software and later releases, you can configure up to eight HTTP outgoing proxy servers. When the server has failed, requests are redirected to the primary HTTP outgoing proxy server. If the primary HTTP outgoing proxy server fails, the request is redirected to the next server that is configured in the list of outgoing proxy servers.

## Designating a Primary Outgoing FTP Proxy Server

In the ACNS 5.2.1 software and later releases, you can configure up to eight proxy servers for FTP-over-HTTP missed traffic.

### Note

At any one time, the Content Engine uses only one of the configured outgoing FTP-over-HTTP proxy servers. They cannot be used simultaneously.
Chapter 14  Configuring Primary and Backup Proxy Servers for Standalone Content Engines

Designating a Primary Outgoing HTTPS Proxy Server

In the ACNS 5.1.x software and earlier releases, you could only configure the Content Engine to use one outgoing HTTPS proxy server. In the ACNS 5.2 software and later releases, you can configure up to eight HTTPS outgoing proxy servers for each Content Engine.

At any one time, the Content Engine uses only one of the configured outgoing HTTPS proxy servers; they cannot be used simultaneously.

To configure a standalone Content Engine to direct all HTTPS miss traffic (HTTPS-over-HTTP) to a parent cache without using ICP or WCCP, you must explicitly designate a proxy server as the primary outgoing HTTPS proxy server for the Content Engine.

To designate a proxy server as the primary outgoing HTTPS proxy server for the Content Engine, use the **https proxy outgoing port primary** global configuration command:

- *host* is the hostname or IP address of the parent cache (outgoing HTTPS proxy server) to which HTTPS missed traffic is directed.
- *port* is the port number used by the parent cache to accept missed HTTPS-over-HTTP requests from the Content Engine.

To set the specified host as the primary outgoing HTTPS proxy server, use the **primary** keyword. If several servers (hosts) are configured with the **primary** keyword, the last one configured becomes the primary outgoing HTTPS proxy server for the Content Engine.

In this example, host 10.1.1.1 on port 8088 is explicitly designated as the primary outgoing HTTPS proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing HTTPS-over-HTTP proxy server:

```
ContentEngineA(config)# https proxy outgoing host 10.1.1.1 8088 primary
ContentEngineA(config)# https proxy outgoing host 10.1.1.2 220
```

**Note**

In the ACNS 5.1 software release, the **ftp proxy outgoing host** command was replaced with the **ftp-over-http proxy outgoing host** command. This change to the CLI command was made to clarify that the command applies only to FTP-over-HTTP requests and does not apply to FTP native requests.

Designating a Primary Outgoing FTP-over-HTTP Proxy Server

To configure a standalone Content Engine to direct all FTP-over-HTTP miss traffic to a parent cache without using ICP or WCCP, you must explicitly designate the parent cache as the primary outgoing FTP-over-HTTP proxy server for the Content Engine.

To designate a proxy server as the primary outgoing FTP proxy server for the Content Engine, use the **ftp-over-http proxy outgoing host host port primary** global configuration command:

- *host* is the hostname or IP address of the parent cache (the outgoing FTP proxy server) to which FTP-over-HTTP missed traffic is directed.
- *port* is the port number used by the parent cache to accept missed FTP-over-HTTP requests from the Content Engine.

To set the specified host as the primary outgoing FTP proxy server, use the **primary** keyword. If several servers (hosts) are configured with the **primary** keyword, the last one configured becomes the primary outgoing FTP-over-HTTP proxy server for the Content Engine.

In this example, host 10.1.1.1 on port 8088 is explicitly designated as the primary outgoing FTP-over-HTTP proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing FTP-over-HTTP proxy server:

```
ContentEngineA(config)# ftp-over-http proxy outgoing host 10.1.1.1 8088 primary
ContentEngineA(config)# ftp-over-http proxy outgoing host 10.1.1.2 220
```
In this example, Content Engine A is configured to send its missed HTTPS traffic (that is, cache misses for browser requests for HTTPS content [HTTPS-over-HTTP requests]) to the host 10.1.1.1 on port 8088. Host 10.1.1.1 is explicitly designated as the primary outgoing HTTPS proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing HTTPS proxy server for Content Engine A.

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

### Configuring HTTP and HTTPS Outgoing Proxy Exclusion Settings

Some situations involve the deployment of a Content Engine in proxy mode at company headquarters and Content Engines in transparent mode at remote locations in branch offices. In this situation, if a cache miss occurs at the remote Content Engine, company policy requires that the request be routed to the Content Engine at headquarters.

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` global configuration command was entered. If this command was not entered, then the Content Engine forwards the request directly to the origin server.

When the Content Engine is operating in transparent mode, it can intercept requests that were sent to another proxy server 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 origin server itself, or if it is configured to use an outgoing proxy server for this protocol, it forwards the request to the specified outgoing proxy server. In this situation, 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 proxy server that the client originally addressed the request to. This may be different from the Content Engine’s own outgoing proxy server for the specified protocol.

The ACNS 5.x software also has an option that allows you to specify a single domain name, hostname, or IP address to be globally excluded from proxy forwarding. The wildcard character * (asterisk) can be used for IP addresses (for instance, 172.16.*.*).

**Note** Requests with a destination specified with wildcard characters bypass the Content Engine proxy as well as the failover proxies.

The Content Engine addresses the request to the destination server directly and not to the client’s intended proxy server.

When a Content Engine intercepts a proxy request intended for another proxy server and there is no outgoing proxy server configured for HTTPS-over-HTTP, and the `proxy-protocols transparent default-server` global configuration command is configured, the Content Engine addresses the request to the destination server directly and not to the client’s intended proxy server.

However, all transparently intercepted requests sent by clients are returned to the client and requested objects are not delivered if the following two conditions exist:

- The `proxy-protocols transparent reset` command is configured on the Content Engine.
- A cache miss occurs.
You can use the Content Engine GUI or the CLI to configure HTTP and HTTPS outgoing proxy exclusion settings.

- From the Content Engine GUI, choose **Caching > Proxy Protocols**. Use the displayed Proxy Protocols window to configure these settings for this standalone Content Engine. For more information about how to use the Proxy Protocols window, click the **HELP** button in the window.
- From the Content Engine CLI, use the **proxy-protocols** global configuration commands. See **Table 14-1** and **Table 14-2**. The order in which the CLI commands are entered is not important.

### Table 14-1 Proxy Protocols Key Parameters

<table>
<thead>
<tr>
<th>Key Content Engine GUI Parameter</th>
<th>Description</th>
<th>Corresponding Content Engine CLI Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default server</td>
<td>Specifies that the Content Engine should retrieve objects from the origin server itself. With this option, a proxy-style request can be sent to an outgoing proxy server if such a server is configured.</td>
<td>proxy-protocols transparent default-server</td>
</tr>
<tr>
<td>Original Proxy</td>
<td>Specifies that the Content Engine should forward the request to the original proxy addressed in the client request.</td>
<td>proxy-protocols transparent original-proxy</td>
</tr>
<tr>
<td>Do not use Outgoing Proxy for the following domains</td>
<td>Excludes the domain name, hostname, or IP address specified here from proxy forwarding.</td>
<td>proxy-protocols outgoing proxy-exclude</td>
</tr>
</tbody>
</table>

To specify a domain name, hostname, or IP address to be excluded from proxy forwarding, use the **proxy-protocols** global configuration command. 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.

```bash
proxy-protocols outgoing-proxy exclude {enable | list word}
proxy-protocols transparent {default-server | original-proxy | reset}
```

**Table 14-2** describes the parameters for the **proxy-protocols** command.

### Table 14-2 Parameters for the proxy-protocols Command

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outgoing-proxy exclude</td>
<td>Sets global outgoing proxy exclude criteria.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables global outgoing proxy exceptions.</td>
</tr>
<tr>
<td>list</td>
<td>Sets the global outgoing proxy exclude list.</td>
</tr>
<tr>
<td>word</td>
<td>Domain names, hostnames, or IP addresses to be excluded from proxy forwarding (supports 64 exclude list entries).</td>
</tr>
<tr>
<td>transparent</td>
<td>Sets transparent mode behavior for proxy requests.</td>
</tr>
<tr>
<td>default-server</td>
<td>Uses the Content Engine to go to the origin server or the outgoing proxy server, if configured.</td>
</tr>
<tr>
<td>original-proxy</td>
<td>Uses the intended proxy server from the original request.</td>
</tr>
<tr>
<td>reset</td>
<td>Resets the incoming connection.</td>
</tr>
</tbody>
</table>
The `proxy-protocols outgoing-proxy exclude` option allows you to specify a single domain name, hostname, or IP address to be globally excluded from proxy forwarding. For example, if you enter the domain name cisco.com, then the configured outgoing proxy server will be bypassed each time the Content Engine tries to retrieve a web page from cisco.com. You can specify IP addresses instead of domain names. The wildcard character (*) can also be specified for IP addresses (for instance, 174.12.*.*). You must press the Enter key after entering each local domain.

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 proxy servers.

When you enter the `proxy-protocols transparent default-server` global configuration command, the Content Engine forwards intercepted HTTP, HTTPS-over-HTTP, and FTP-over-HTTP requests to the corresponding outgoing proxy server, if one is configured on the Content Engine. 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 directed back to the intended proxy server.

The `proxy-protocols transparent reset` option specifies that requests sent by a web client to another proxy server, but intercepted by the Content Engine in transparent mode, be returned to the web client during a cache miss. The requested objects are not delivered.

The following example configures the Content Engine to forward intercepted HTTPS-over-HTTP requests to an outgoing proxy server. The domain name cruzio.com is excluded from proxy forwarding.

```
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 list cruzio.com
```

To verify the configuration, enter the `show proxy-protocols` EXEC command.

```
ContentEngine# show proxy-protocols all
Transparent mode forwarding policies: default-server
Outgoing proxy exclude list is enabled
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
```

### Monitoring Outgoing Proxy Servers and Statistics

A background process on the Content Engine monitors the state of the configured outgoing proxy servers. You can configure the Content Engine to poll the specified outgoing proxy servers at a specific interval in order to monitor their availability.

This monitor interval is the frequency which the proxy servers are polled. The monitoring interval is specified in seconds, and can be from 10 to 300 seconds. The default monitoring interval is 60 seconds. If one of the outgoing proxy servers is unavailable, the polling mechanism waits for the connect timeout (300000 microseconds) before polling the next outgoing proxy server.

Use the following global configuration commands to specify the monitoring interval:

```
ContentEngine(config)# proxy-protocols monitoring interval <seconds>
ContentEngine(config)# proxy-protocols monitoring timeout <microseconds>
```
Displaying the Current Outgoing Proxy Server Configuration

To display the Content Engine’s current outgoing proxy server configuration, use the following EXEC commands:

- To display the current outgoing HTTP proxy server configuration, enter the `show http proxy` command.
- To display the current outgoing HTTPS proxy server configuration, enter the `show https proxy` command.
- To display the current outgoing FTP-over-HTTP proxy server configuration, enter the `show ftp-over-http` command. (In the ACNS 5.3.1 software release, the `show ftp proxy` command was replaced with the `show ftp-over-http` and `show ftp-native` commands.)

Displaying Outgoing Proxy Server Statistics

To display statistics about the HTTP requests that the Content Engine has sent to the specified HTTP proxy server, enter the `show statistics http proxy outgoing` EXEC command.

To display statistics about the FTP-over-HTTP requests that this Content Engine has sent to the specified FTP-over-HTTP proxy server, enter the `show statistics ftp-over-http` EXEC command.

**Note**

In the ACNS 5.3.1 software release, the `show statistics ftp` EXEC command was replaced with the `show statistics ftp-over-http` and `show statistics ftp-native` EXEC commands.
Configuring Advanced Transparent Caching Features on Standalone Content Engines

This chapter describes how to configure advanced transparent caching features on standalone Content Engines that are running the ACNS 5.4.x software and later releases. The chapter includes the following sections:

- About Advanced Transparent Caching Features, page 15-2
- Configuring Bypass Settings on Standalone Content Engines, page 15-3
- Configuring WCCP Flow Protection, page 15-9
- Configuring WCCP Slow Start, page 15-10
- Configuring WCCP IP Spoofing, page 15-10
About Advanced Transparent Caching Features

One of the fundamental principles of transparent network caching is that the Content Engine 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 ACNS software uses a WCCP-enabled router and various advanced techniques to ensure that the Content Engine remains transparent, even if client browsers are nonoperational or web servers are not HTTP-compliant.

Table 15-1 lists the advanced transparent caching features. This set of features ensures that you do not encounter unexpected problems when you deploy the Cisco caching solutions (including the deployment of standalone Content Engines as caching engines).

Table 15-1  Advanced Transparent Caching Features for Standalone Content Engines

<table>
<thead>
<tr>
<th>Technology Service</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bypass</td>
<td></td>
</tr>
<tr>
<td>Authentication traffic bypass</td>
<td>Preserves cache transparency and avoids disruption of service by allowing the Content Engine to generate a bypass list for the selected client/server pairs. For more information, see the “Configuring Authentication Traffic Bypass on Standalone Content Engines” section on page 15-4.</td>
</tr>
<tr>
<td>Static bypass</td>
<td>Permits traffic from the specified sources to bypass the Content Engine. For more information, see the “Configuring Static Bypass on Standalone Content Engines” section on page 15-7.</td>
</tr>
<tr>
<td>Overload bypass</td>
<td>Prevents a Content Engine from becoming a bottleneck when traffic loads exceed the capacity of the Content Engine. For more information, see the “Configuring Overload Bypass on Standalone Content Engines” section on page 15-8.</td>
</tr>
<tr>
<td>WCCP flow protection</td>
<td>Prevents existing flows from being broken when the WCCP cluster load distribution changes because of the addition or subtraction of a Content Engine into or from a cluster. For more information, see the “Configuring WCCP Flow Protection” section on page 15-9.</td>
</tr>
<tr>
<td>WCCP slow start</td>
<td>Prevents cluster destabilization when a new Content Engine is added to a heavily loaded cluster. For more information, see the “Configuring WCCP Slow Start” section on page 15-10.</td>
</tr>
<tr>
<td>WCCP IP spoofing</td>
<td>Uses the client IP address when connecting to the origin web server. For more information, see the “Configuring WCCP IP Spoofing” section on page 15-10.</td>
</tr>
</tbody>
</table>

The ACNS software also has a built-in bypass feature that is not configurable. This built-in bypass feature affects connections that are opened before WCCP is activated, and affects retransmissions of client packets after the connection has been terminated. Through this built-in bypass feature, the Content Engine automatically sends such traffic back to the router.

The following sections in this chapter describe how to configure these advanced transparent caching features on standalone Content Engines:

- Configuring Bypass Settings on Standalone Content Engines, page 15-3
- Configuring WCCP Slow Start, page 15-10
- Configuring WCCP Flow Protection, page 15-9
Chapter 15      Configuring Advanced Transparent Caching Features on Standalone Content Engines

Configuring Bypass Settings on Standalone Content Engines

Bypass refers to a method that a Content Engine can use to handle various error responses (including authentication failure) from an origin server. When the Content Engine receives an error response from an origin server, it adds an entry for the server to its bypass list. When it receives subsequent requests for content residing on the bypassed server, it redirects packets to the bypass gateway. If no bypass gateway is configured, then the packets are returned to the redirecting Layer 4 switch.

Bypass features can be used with a WCCP Version 2 router or with a Layer 4 switch, such as the Cisco Content Switching Module or the Content Services Switch (CSS) switch.

This section describes how to configure a standalone Content Engine to support the following types of bypass:

- Configuring Authentication Traffic Bypass on Standalone Content Engines, page 15-4
- Configuring Static Bypass on Standalone Content Engines, page 15-7
- Configuring Overload Bypass on Standalone Content Engines, page 15-8

Note: The bypass feature is only available when WCCP Version 2 is enabled in your local network. The Content Engine can only bypass WCCP-redirected traffic and not proxy-style requests.

To display a bypass summary that includes the number of entries in the bypass list, enter the show bypass summary EXEC command:

ContentEngine# show bypass summary
Total number of requests bypassed = 0
  Requests bypassed due to system overload = 0
  Requests bypassed due to authentication issues = 0
  Requests bypassed due to facilitate error transparency = 0
  Requests bypassed due to static configuration = 0
Total number of entries in the bypass list = 1
  Number of Authentication bypass entries = 0
  Number of Error bypass entries = 0
  Number of Static Configuration entries = 1
L2 Bypass:
  Number of L2 bypassed packets = 0
ContentEngine#

To display a list of entries in the bypass list (see sample output below), enter the show bypass list EXEC command:

ContentEngine# show bypass list

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
<th>Entry type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.1:0</td>
<td>5.5.5.5:0</td>
<td>static-config</td>
</tr>
</tbody>
</table>
Configuring Authentication Traffic Bypass on Standalone Content Engines

Occasionally a website uses the client’s IP address to authenticate a client. This method of client authentication is typically used only in older origin web servers. However, in such situations there must be a way for the Content Engine “to get out of the way” between the client and the origin web server so that the client can be authenticated by the origin web server. With direct proxy routing, this is not possible because the client is configured to point directly to the Content Engine as its outgoing proxy server (the Content Engine receives the client requests directly from the client browser or media player). However, with WCCP-intercepted requests, if the authentication traffic bypass feature is enabled on the Content Engine, then WCCP-intercepted requests can bypass the Content Engine so that the origin web server can authenticate the client.

Another typical use of this bypass feature, occurs with websites that do not allow the Content Engine to connect directly on behalf of the client because of IP authentication. In order to preserve cache transparency and avoid disruption of service, the Content Engine can use this bypass feature to generate a dynamic access list for selected client-server pairs. Authentication traffic bypass triggers are also propagated upstream and downstream in the case of hierarchical caching.

By default, the authentication traffic bypass feature is disabled on a Content Engine. To enable the authentication traffic bypass feature on a standalone Content Engine, use the `bypass auth-traffic` global configuration command. When a client-server pair enters authentication traffic bypass, the pair is bypassed for an amount of time set by the `bypass timer` global configuration command (20 minutes by default). For example, when a client-server pair performs authentication traffic bypass, it is bypassed for a configurable amount of time, which is set by the `bypass timer` global configuration command.

If the authentication traffic bypass feature is enabled on a Content Engine, then the following occurs:

1. The client (an end user who is using a browser to request content) sends a content request to a web server (an origin web server).
   
   **Note** Only WCCP redirected requests can be bypassed. With proxy-style requests, client browsers are explicitly configured to point directly to the Content Engine as their proxy server; therefore, bypass cannot be configured for proxy-style requests.

2. A WCCP router transparently intercepts the content request and forwards it to the Content Engine (transparent proxy server).
3. The Content Engine masquerades as the origin web server and responds to the client. At the same time, the Content Engine sends a request to the origin web server using its own IP address.
4. If the origin web server is performing any kind of request authentication that is based on IP addresses, it rejects the request.
5. If the Content Engine receives a 401, 403, 501, 503, 502, 503, 504, or 505 response from the origin web server, it performs the following authentication traffic bypass actions:
   a. It sends a redirect to the client with the exact same URL.
   b. It adds a bypass entry (the client-server pair entry) to the bypass list.
6. The retried request from the client is intercepted again by the WCCP Version 2 router and is forwarded to the Content Engine. This time, the Content Engine forwards the request to the origin web server instead of handling the request itself because of the client-server bypass entry that was just created in the bypass list.
7. The origin web server responds to the client, and this response goes directly to the client.
You can use either the Content Engine GUI or the CLI to configure authentication traffic bypass on standalone Content Engines:

- From the Content Engine GUI, choose **Caching > Bypass**. The Bypass window appears. Click the **Authentication Bypass On** radio button to enable authentication traffic bypass. In the Bypass Entry Expiration Time field, specify a value (in minutes) to set the number of minutes that an idle client-server pair remains on the bypass access list. The default value is 20 minutes. Click **Update** to save the settings.

- From the Content Engine CLI, use the `bypass auth-traffic` and `bypass gateway` global configuration commands. The parameters are described in **Table 15-2**.

### Table 15-2 Authentication Traffic Bypass Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth-traffic</td>
<td>Sets the authenticated traffic bypass feature configuration.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables authentication traffic bypass.</td>
</tr>
<tr>
<td>gateway</td>
<td>Configures a router to which bypassed packets are redirected when the Content Engine receives requests redirected by a Layer 4 switch.</td>
</tr>
<tr>
<td>ipaddress</td>
<td>IP address of the router acting as the bypass gateway.</td>
</tr>
<tr>
<td>timer</td>
<td>Sets the authentication bypass timer (in minutes). The bypass entry is removed from the dynamic list when the timer expires.</td>
</tr>
<tr>
<td>minutes</td>
<td>Time in minutes (1–1440).</td>
</tr>
</tbody>
</table>

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
```

To identify the WCCP Version 2 router to which the Content Engine will direct responses when errors are received from the origin server, use the `bypass gateway ipaddress` global configuration command. Replace `ipaddress` with the IP address of a router that is a Layer 2 neighbor of the Content Engine.

To enable bypass with a Layer 4 switch, use the `http l4 switch enable` global configuration command.

To disable the authentication traffic bypass feature on a standalone Content Engine, use the **no** form of the `bypass auth-traffic` global configuration command.

**Note**

The `bypass auth-traffic` global configuration command is also used to enable transparent error handling on standalone Content Engines.

### Example 1—Dynamic Bypass upon Receiving a Web Server Error

This example and the one following implement the WCCP return-path functionality, which enables a Content Engine to 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.

About 3 percent of all HTTP traffic flows will 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.
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 (a cache miss), and issues a request for the object from the origin web server, but receives some kind of error message (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 has failed, but it does not know the cause (for example, the origin web server is performing authentication based on user source IP address, or there is an incompatibility between the TCP stacks).

Dynamic client bypass in this case means that the Content Engine returns the HTTP response code to the browser. The response returned is an HTML page with a meta tag, requesting a refresh. 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, which bypasses the Content Engine. (See Figure 15-1.)

![Figure 15-1 Dynamic Traffic Bypass](image)

**Example 2—Dynamic Bypass upon Receiving an Unsupported Protocol**

When the Content Engine receives non-HTTP requests over TCP port 80, it issues a retry response, closes the connection, and does not accept subsequent connections just as it does as in Example 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.
Not all HTTP clients support HTML. If the client or server does not support HTTP or HTML, then the client can experience problems and the Content Engine will not be able to serve the requested content to the client.

**Configuring Static Bypass on Standalone Content Engines**

The static bypass feature permits traffic from specified sources to bypass a Content Engine. The types 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

To enable and configure the static bypass feature on a standalone Content Engine, use the `bypass static` global configuration command:

```
bypass static {clientip | any-client} {serverip | any-server}
```

Table 15-3 describes the command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>Adds a static entry to the bypass list.</td>
</tr>
<tr>
<td>clientip</td>
<td>IP address from which requests will bypass the Content Engine. Wildcards are not supported.</td>
</tr>
<tr>
<td>serverip</td>
<td>IP address to which requests will bypass the Content Engine. Wildcards are not supported.</td>
</tr>
<tr>
<td>any-server</td>
<td>Requests from a specified client to any server will bypass the Content Engine.</td>
</tr>
<tr>
<td>any-client</td>
<td>HTTP traffic from any client destined to a particular server will bypass the Content Engine.</td>
</tr>
</tbody>
</table>

To clear all static configuration lists on a Content Engine, use the `no` form of the `bypass static` global configuration command.

The following are some examples of how to use the `bypass static` global configuration command to configure the static bypass feature on a standalone Content Engine.

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

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

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

```
ContentEngine(config)# bypass static 10.1.17.1 any-server
```
Chapter 15 Configuring Advanced Transparent Caching Features on Standalone Content Engines

Configuring Overload Bypass on Standalone Content Engines

When a Content Engine is overloaded and the overload bypass option is enabled, the Content Engine bypasses a bucket and reroutes the overload traffic. If the load remains too high, another bucket is bypassed, and so on, until the Content Engine can handle the load. (See Figure 15-2.)

Figure 15-2 Overload Bypass

A bucket is defined as a certain subsection of the allotted hash assigned to each Content Engine in a Content Engine cluster. If only one Content Engine exists in this environment, it has 256 buckets assigned to it.

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

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

You can use the Content Engine GUI or the CLI to configure overload bypass on a standalone Content Engine:

- From the Content Engine GUI, choose Caching > Bypass, and use the displayed Bypass window. For more information about how to use the Bypass window to configure load bypass, click the HELP button in the Bypass window.
- From the Content Engine CLI, use the bypass load global configuration command:

  bypass load { enable | in-interval seconds | out-interval seconds | time-interval minutes }

Table 15-4 describes the command parameters.

Table 15-4 bypass load Command Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>load</td>
<td>Adds a static entry to the bypass list.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables overload bypass on the Content Engine.</td>
</tr>
</tbody>
</table>
Chapter 15    Configuring Advanced Transparent Caching Features on Standalone Content Engines

Configuring WCCP Flow Protection

The WCCP flow protection feature ensures that no existing flows are broken when a new Content Engine is brought online. When transparent traffic interception or redirection first begins, WCCP flow protection ensures that no existing HTTP flows are broken by allowing preexisting, established HTTP flows 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 Content Engines 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.

To implement WCCP flow protection, use the `wccp flow-redirect` global configuration command. This command works with WCCP Version 2 only. Flow protection is designed to keep the TCP flow intact as well as to not overwhelm Content Engines when they are first started up or are reassigned new traffic. This feature also has a slow start mechanism that enables Content Engines to attempt to take a load that is appropriate for their capacity.

This example shows how to enable WCCP flow protection on a standalone Content Engine:

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

When bypass is enabled, the client itself tries to reach the origin web server. You must disable all bypass options to eliminate an unnecessary burden on the network.

In the ACNS 5.3.1 software and later releases, enter the `show wccp flows ftp-native` EXEC command to display summary information about the native FTP caching service packet flows.
Configuring WCCP Slow Start

Within a cluster of Content Engines, TCP connections are redirected to other Content Engines as units are added or removed. A Content Engine can be overloaded if it is reassigned new traffic too quickly or if it is introduced abruptly into a fat pipe.

WCCP slow start performs the following tasks in order to prevent a Content Engine from being overwhelmed when it comes online or is reassigned new traffic:

- TCP flow protection when WCCP Version 2 is enabled and a Content Engine is introduced into the cluster
- TCP flow protection when WCCP Version 2 is disabled and a Content Engine is leaving the cluster
- Load assignment to the Content Engine in slow increments rather than a full load at boot up

Slow start is applicable only in the following cases:

- Initial boot up when there is no Content Engine yet present in the server farm.
- When a new Content Engine is added to a cluster that is not handling the full load; for example, when there are some buckets that are being shed by the cluster.

In all other cases slow start is not necessary, and all the Content Engines can be assigned their share of traffic right away.

To enable the slow start capability of the caching service on a standalone Content Engine, enter the `wccp slow-start enable` global configuration command. To disable slow start capability, enter the `no` form of this command.

In the ACNS 5.3.1 software and later releases, enter the `show wccp slowstart ftp-native` EXEC command to display WCCP slow start information on standalone Content Engines.

Configuring WCCP IP Spoofing

With typical transparent caching, an end user issues an HTTP request from a web browser. This request is transparently intercepted and redirected to the Content Engine (acting as a transparent proxy server) by a WCCP router. The Content Engine accepts the incoming TCP connection from the WCCP router, determines that the request is for an object not in storage (cache miss), and issues a request to the origin server for the requested object. When the Content Engine contacts the origin server, it uses its own IP address instead of the IP address of the client for which it is making the request.

If IP spoofing is configured on the WCCP Version 2-enabled routers and the Content Engines, the Content Engine (acting as a transparent proxy server) can send out the client’s IP address to the origin server for authentication purposes instead of sending out the request with its own IP address. The WCCP router can also intercept packets from the server that are destined for the client’s IP address, and redirect these packets to the Content Engine.
By spoofing a client’s IP address, the following capabilities are supported:

- The Content Engine can send out packets with the client IP (which is different from the Content Engine’s own IP address).
- The Content Engine can receive packets with the client IP (which is again different from the Content Engine’s own IP address), and send the packet to the correct application that is waiting for the packet.
- The WCCP Version 2-enabled router can intercept the packets from both the client and the server transparently, and forward these redirected packets to the same Content Engine so that the TCP connection is not broken.

With the ACNS software earlier than the ACNS 5.0.7 software release, IP address spoofing for transparently intercepted proxy-style requests was not supported. However, with the ACNS 5.0.7 software and later releases, IP address spoofing is performed for transparently intercepted proxy-style requests when the Content Engine is configured to use the proxy server from the original request to retrieve the content. To configure a standalone Content Engine to use the proxy server from the original request, use the `proxy-protocols transparent original-proxy` global configuration command.

With a proxy-style request, the client sends a proxy-style HTTP request if the client is configured to send HTTP requests directly to a specific Content Engine. The client sends the request to the IP address of the proxy server, with the complete destination URL, including the name of the origin server in the HTTP method (for example, GET).

In the case of a server-style HTTP request, the client sends the request directly to the destination server with the HTTP Host header containing the domain name of the origin server, and the HTTP method containing the path to the file or script that the client is requesting.

The proxy-style request for myfile.html located in mydirectory in the domain myserver.com, when transparently intercepted, will have the following initial HTTP line:

```
GET http://myserver.com/mydirectory/myfile.html HTTP/1.1
```

The server-style request for myfile.html located in mydirectory in the domain myserver.com, when transparently intercepted, will have the following initial HTTP line:

```
GET /mydirectory/myfile.html HTTP/1.1
```

To enable IP spoofing on a standalone Content Engine, enter the `wccp spoof-client-ip enable` global configuration command. To disable IP spoofing, enter the `no wccp spoof-client-ip enable` global configuration command.

---

**Tip**

The Content Engine can also use authentication traffic bypass to automatically generate a dynamic access list and to connect to a server using the client’s IP address for selected client-server pairs. For more information on this topic, see the “Configuring Authentication Traffic Bypass on Standalone Content Engines” section on page 15-4. You can also forward the client’s IP address without turning IP spoofing on by using the `http append x-forwarded-for-header` global configuration command on the Content Engine that is serving the request.

IP spoofing is recommended in the following situations:

- Logging of user IP addresses
- Filtering based on user IP addresses
- Policy-based routing to provide some users better service than others
Because IP spoofing can break in various subtle situations, to avoid potential problems for ACNS users several restrictions have been put in place. These restrictions block IP spoofing of some or all client traffic even though globally IP spoofing is enabled. IP spoofing is intentionally avoided in the following situations:

- The client request is not transparently redirected (meaning, proxy-style requests)
- The request is transparently redirected but the request is proxy style, and the proxy-protocols transparent original-proxy configuration is not present
- An HTTP outgoing proxy has been configured
- The client request matches any of the following rule patterns:
  - rule use-proxy
  - rule use-server
  - rule rewrite

Examples of Configuring IP Spoofing with Standalone Content Engines

To configure standalone Content Engines and WCCP Version 2-enabled routers for IP spoofing, you must configure both the Content Engine and WCCP Version 2-enabled routers for IP spoofing.

**Note**

Before you can enable IP spoofing on a Content Engine, you must configure the router interfaces serving the client, the origin server, and the Content Engine. The clients, Content Engines, and origin servers must be configured on three separate interfaces on the WCCP Version 2-enabled router.

This section provides some examples of how to configure IP spoofing with standalone Content Engines and WCCP Version 2-enabled routers:

- Example 1—IP Spoofing with the Content Engine and Clients on Different Subnets, page 15-13
- Example 2—IP Spoofing with a Reverse Proxy Server, page 15-15
Example 1—IP Spoofing with the Content Engine and Clients on Different Subnets

This section provides an example of how to configure a standalone Content Engine and WCCP Version 2 router for IP spoofing. In this example, the Content Engine and the requesting clients are on different subnets, and two WCCP services (the web-cache service and service 95) are configured. One WCCP service (the web-cache service) hashes on the destination IP address, and the other WCCP service (service 95) hashes on the source IP address. (See Figure 15-3.)

Figure 15-3  IP Spoofing with Content Engine and Clients on Different Subnets

![IP Spoofing Diagram]

Note

The custom-web-cache service (service 98) could also be used instead of the standard web-cache service (service 0) for the WCCP service that hashes on the destination IP address. For a list of WCCP services, see Table B-3.

To configure IP spoofing on a standalone Content Engine and a WCCP Version 2-enabled router when the Content Engine and clients are on different subnets, follow these steps:

**Step 1** Enable WCCP Version 2 on the Content Engine, which is on a different subnet than the clients.

```
ContentEngine# configure terminal
ContentEngine(config)# wccp version 2
```

**Step 2** On the Content Engine, configure a router list. In this case, the router list number 1 is created and it has only a single WCCP router, which has an IP address of 10.10.20.1.

```
ContentEngine(config)# wccp router-list 1 10.10.20.1
```

**Step 3** On the Content Engine, configure port list 1 to be associated with a WCCP service through port 80.

```
ContentEngine(config)# wccp port-list 1 80
```
Chapter 15  Configuring Advanced Transparent Caching Features on Standalone Content Engines

Configuring WCCP IP Spoofing

Step 4  On the Content Engine, configure a user-defined WCCP service (service 95) that hashes on the source IP address by using the `wccp service-number` global configuration command.

   a. Specify the WCCP service number.

   b. Associate this service with the list of WCCP Version 2-enabled routers (router list number 1) and ports (port list number 1) that will be used to support this WCCP service.

   c. Associate the hashing parameters with the source IP address and the source port.

   d. Specify the `ip match-source-port` option that hashes on the source IP address.

   ```
   ContentEngine(config)# wccp service-number 95 router-list-num 1 port-list-num 1 application cache hash-source-ip match-source-port
   ```

   **Note**  WCCP services number 90 to 97 are for user-defined services, as described in Table B-3. A user-defined service is a WCCP service in which port numbers can be configured to redirect traffic to a Content Engine. In this scenario, service 95 is used to create a user-defined WCCP service. By specifying the `application cache` option of the `wccp service-number` global configuration command, the traffic is redirected to the Content Engine’s conventional caching processes (for HTTP requests), and the `application streaming` option redirects traffic to the Content Engine’s media caching processes (for WMT or RTSP requests).

Step 5  Inform the WCCP router that the Content Engine is accepting redirected web traffic.

   ```
   ContentEngine(config)# wccp web-cache router-list-num 1
   ```

Step 6  Enable client IP spoofing on the Content Engine.

   ```
   ContentEngine(config)# wccp spoof-client-ip enable
   ```

Step 7  Exit global configuration mode on the Content Engine.

   ```
   ContentEngine(config)# exit
   ```

Step 8  Write the running configuration to nonvolatile memory.

   ```
   ContentEngine# write memory
   ```

   Now that IP spoofing in configured on the Content Engine, complete the remaining steps to configure the WCCP Version 2 router for IP spoofing.

Step 9  Ensure that WCCP Version 2 is enabled on the router.

   ```
   Router(config)# ip wccp version 2
   ```

Step 10  Instruct the WCCP router to run the web-cache service (service 95).

   ```
   Router(config)# ip wccp web-cache
   ```

Step 11  Enable service 95 on the WCCP router.

   ```
   Router(config)# ip wccp 95
   ```

Step 12  Specify an interface to configure, and enter interface configuration mode.

   ```
   Router(config)# interface type number
   ```

Step 13  Enable WCCP redirection on the WAN interface (the router interface that is connected to the origin server) with the service that hashes on the destination IP address (the web-cache service). (See Figure 15-3.)

   ```
   Router(config-if)# ip wccp web-cache redirect out
   ```
Step 14  Enable WCCP redirection on the LAN interface (the router interface that is connected to the client) with the service that hashes on the source IP address (service 95). (See Figure 15-3.)

Router(config-if)# ip wccp 95 redirect out

Step 15  Disable traffic redirection on the router interface that is connected to the Content Engine. (See Figure 15-3.)

Router(config-if)# ip wccp redirect exclude in

You must disable traffic redirection on the router interface that is connected to the Content Engine to avoid loopbacks as the WCCP router tries to send the packet with the source IP address back to the Content Engine.

Step 16  Exit global configuration mode.

Router(config-if)# exit

Step 17  Save the configuration on the router.

Router# copy running-config startup-config

---

Example 2—IP Spoofing with a Reverse Proxy Server

This section provides an example of how to configure a standalone Content Engine and WCCP Version 2 router for IP spoofing. In this example, the Content Engine is functioning as a reverse proxy server. (See Figure 15-4.)

**Figure 15-4  IP Spoofing with a Reverse Proxy Server**
To configure IP spoofing on a standalone Content Engine (reverse proxy server) and a WCCP Version 2-enabled router, follow these steps:

**Step 1** Enable WCCP Version 2 on the Content Engine (that is acting as a reverse proxy server).

```
ContentEngine(config)# wccp version 2
```

**Step 2** Configure a router list on the Content Engine. In this case, there is only a single router (a WCCP Version 2-enabled router that has an IP address of 10.10.20.1) on router list number 1.

```
ContentEngine(config)# wccp router-list 1 10.10.20.1
```

**Step 3** Associate port list 1 with a WCCP service through port 80.

```
ContentEngine(config)# wccp port-list 1 80
```

**Step 4** Configure a user-defined WCCP service (service 96) on the Content Engine. Associate the router list, port list, and hashing parameters with the destination IP address and the source port for this WCCP service.

```
ContentEngine(config)# wccp service-number 96 router-list-num 1 port-list-num 1 application cache hash-destination-ip match-source-port
```

**Note** If you have a Content Engine cluster and you are using weight assignments within this cluster, you must make sure that the weight assignments for the service groups assigned to IP spoofing for both outbound and inbound packets are equal on all Content Engines to prevent a break in the TCP connection. Use the `wccp service-number servnumber router-list-num num port-list-num port application cache weight percentage` command to establish a weight for these service groups if needed. By default, the Content Engine cluster hashes appropriately with IP spoofing turned on, so assigning weights to service groups is not needed.

**Step 5** Inform the WCCP Version 2-enabled routers in the specified router list (for example, router list number 1) that the Content Engine (that is acting as a reverse proxy server) is accepting HTTP reverse proxy traffic.

```
ContentEngine(config)# wccp reverse-proxy router-list-num 1
```

**Step 6** Enable client IP spoofing on the Content Engine.

```
ContentEngine(config)# wccp spoof-client-ip enable
```

**Step 7** Exit global configuration mode.

```
ContentEngine(config)# exit
```

**Step 8** Write the running configuration to nonvolatile memory.

```
ContentEngine# write memory
```
Step 9  Configure the router for IP spoofing, as follows:
   a. Enable the reverse-proxy service (service 99).
      
      `Router(config)# ip wccp 99`
   
   b. Specify an interface to configure, and enter interface configuration mode.
      
      `Router(config)# interface type number`
   
   c. Configure redirect out for WCCP service number 96 (the service that hashes on the destination IP address) on the interface that is connected to the client.
      
      `Router(config-if)# ip wccp 96 redirect out`
   
   d. Configure redirect out for the reverse-proxy service (service 99) on the interface that is connected to the origin server.
      
      `Router(config-if)# ip wccp 99 redirect out`

Step 10 Disable WCCP redirection on the router interface that is connected to the Content Engine (that is acting as a reverse proxy server).

   `Router(config-if)# ip wccp redirect exclude in`

Step 11 Exit global configuration mode.

   `Router(config-if)# exit`
Chapter 15 Configuring Advanced Transparent Caching Features on Standalone Content Engines

Configuring WCCP IP Spoofing
CHAPTER 16

Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines

This chapter describes how to set up additional network interfaces and configure bandwidth for these interfaces and content services in a locally managed deployment.

This chapter contains the following sections:

- Configuring Additional Network Interfaces, page 16-2
- Defining Interface Descriptions, page 16-10
- Configuring Bandwidth for Interfaces and Content Services, page 16-11

Note

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication.
Configuring Additional Network Interfaces

When you initially configured a standalone Content Engine, you chose an initial interface and either configured it for DHCP, or gave it a static IP address. You can use the Content Engine CLI to configure additional network interfaces on the Content Engine for redundancy, load balancing, and performance optimization.

You can configure multiple network interfaces as either active-active interfaces or as active-standby interfaces. To configure multiple interfaces as active-active, use the interface global configuration command and assign an IP address to each interface. When multiple interfaces are configured on a Content Engine, they are active simultaneously and improve performance.

This section describes how to configure additional interfaces on standalone Content Engines.

About Virtual Interfaces

Port channel and standby group are two types of virtual network interfaces that are supported by standalone Content Engines. For more information, see the following sections:

- Configuring Port Channels (EtherChannel), page 16-2
- Configuring Standby Groups, page 16-4

Configuring Port Channels (EtherChannel)

EtherChannel for the ACNS 5.x software supports the grouping of up to four same-speed network interfaces into one virtual interface. This grouping is called a port channel or channel group. This grouping capability allows the setting or removing of a virtual interface that consists of two, three, or four Fast Ethernet interfaces or two Gigabit Ethernet interfaces.

EtherChannel also provides interoperability with Cisco routers, switches, and other networking devices or hosts supporting EtherChannel; load balancing; and automatic failure detection and recovery based on each interface’s current link status.

To create up to two port channels on a standalone Content Engine, use the interface PortChannel configuration command.
Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines

Chapter 16

Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines

Configuring Additional Network Interfaces

Table 16-1 lists the interface PortChannel command options.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PortChannel</td>
<td>Selects Ethernet channel of interfaces to configure.</td>
</tr>
<tr>
<td>1</td>
<td>Sets the Port Channel interface number to 1.</td>
</tr>
<tr>
<td>2</td>
<td>Sets the Port Channel interface number to 2.</td>
</tr>
<tr>
<td>ip</td>
<td>(Optional) Enables IP configuration commands for the interface.</td>
</tr>
<tr>
<td>access-group</td>
<td>Configures access control for IP packets on this interface using access control lists (ACLs). For more information about ACLs, see Chapter 19, “Creating and Managing IP Access Control Lists for Standalone Content Engines.”</td>
</tr>
<tr>
<td>ip-access list</td>
<td>Numeric identifier that identifies the ACL to apply to the current interface. For standard ACLs, the valid range is 1 to 99; for extended ACLs, the valid range is 100 to 199.</td>
</tr>
<tr>
<td>access-list_name</td>
<td>Alphanumeric identifier of up to 30 characters, beginning with a letter that identifies the ACL to apply to the current interface.</td>
</tr>
<tr>
<td>in</td>
<td>Applies the specified ACL to inbound packets on the current interface.</td>
</tr>
<tr>
<td>out</td>
<td>Applies the specified ACL to outbound packets on the current interface.</td>
</tr>
<tr>
<td>address</td>
<td>Sets the interface IP address.</td>
</tr>
<tr>
<td>ip-address</td>
<td>IP address of this interface.</td>
</tr>
<tr>
<td>netmask</td>
<td>Netmask of this interface.</td>
</tr>
<tr>
<td>shutdown</td>
<td>(Optional) Shuts down this interface.</td>
</tr>
</tbody>
</table>

The following example shows how to create a port channel on a Content Engine. In this example, the port channel is port channel 2 and is assigned an IP address of 10.10.10.10 and a netmask of 255.0.0.0.

```
ContentEngine# configure
ContentEngine(config)# interface PortChannel 2
ContentEngine(config-if)# ip address 10.10.10.10 255.0.0.0
ContentEngine(config-if)# exit
```

To remove (tear down) a port channel, use the no form of the command, as follows:

```
ContentEngine(config)# interface PortChannel 2
ContentEngine(config-if)# no ip address 10.10.10.10 255.0.0.0
ContentEngine(config-if)# exit
```

To add or remove ports from a port channel, use the commands in the following examples. You can use either the Fast Ethernet or the Gigabit Ethernet ports to form a port channel; however, a port channel cannot contain both Fast Ethernet and Gigabit Ethernet interfaces. A physical interface can be added to a port channel depending upon the device configuration.

To add an interface to a port channel:

```
ContentEngine# configure
ContentEngine(config)# interface FastEthernet 1/1
ContentEngine(config-if)# channel-group 2
ContentEngine(config-if)# exit
```
To remove an interface from a port channel:

```
ContentEngine(config)# interface FastEthernet 1/1
ContentEngine(config-if)# no channel-group 2
ContentEngine(config-if)# exit
```

To configure load balancing for a port channel, use the `port-channel load-balance` global configuration command:

```
ContentEngine(config)# port-channel load-balance
```

The following `load-balance` options are available:

- `dst-ip` Destination IP Address
- `dst-mac` Destination MAC Address
- `round-robin` Round robin each interface (default)

Round robin allows traffic to be distributed evenly between all interfaces in the port channel. The other balancing options give you flexibility in choosing interfaces when sending an Ethernet frame. The `load-balance` command is effective globally. If two port channels are configured, they have to use the same load-balancing option.

### Configuring Standby Groups

In the ACNS 5.2.1 software release and later releases, you can configure one or more interfaces to act as a backup interface (a standby interface) for another interface on a standalone Content Engine. This feature is called standby interface support. Standby groups, which are logical groups of interfaces, are used to implement this feature. When an active network interface fails (because of cable trouble, Layer 2 switch failure, high error count, or other failures) and that interface is part of a standby group, a standby interface can become active and take the load off the failed interface.

There must be at least two interfaces in a standby group. Interfaces that are part of a standby group are called `member interfaces`. After you create a standby group, you define which interfaces should be assigned to this logical group. When defining the member interfaces, you specify the priority of each member interface in a standby group. The member interface with the highest assigned priority is the active interface for that particular standby group. If the active interface fails, the operational member interface with the next highest priority in the standby group comes up, and so forth. If all member interfaces of a particular standby group are down and then one of the member interfaces comes up, the ACNS software detects this situation and brings up the standby group on the member interface that just came up.

In the ACNS 5.3.1 software and later releases, the failure or failover of member interfaces within a standby group will trigger alarms and traps (if alarms and traps are enabled on the Content Engine). Alarms are sent out when failover occurs between member interfaces in a standby group. Specifically, minor alarms are sent out when member interfaces fail, and these alarms are cleared automatically when the interface failover has been successfully completed. Major alarms are sent out if the standby group goes down (that is, no member interface in a standby group can be brought up.)

A physical interface can belong to more than one standby group. Consequently, a single interface can act as a standby interface for more than one standby group.

This standby interface feature can also be used to support a redundant network that uses Layer 4 Cisco Content Services Switch (CSS) switches) to load balance requests to multiple Content Engines. The CSS switch supports active-standby configuration. If the active CSS switch fails, the standby CSS switch takes over all of the load. In such a case, the Content Engine detects this failure and starts serving the
same IP address (shared IP address) on the standby network interface card (NIC) and preserves the existing TCP sessions (session-level redundancy). Session-level redundancy is only possible if the CSS switch can preserve sessions in a failure situation. If the CSS switch loses the sessions, the session will be lost.

To configure standby interfaces, interfaces are logically assigned to standby groups. The following rules define the standby group relationships:

- Each standby group is assigned a unique standby IP address, shared by all member interfaces of the standby group. The IP address of the standby group is shared among the member interfaces; however, only the active interface of the standby group uses this shared IP address at any one time. This shared IP address is configured as an alias on the active interface.
- In the ACNS 5.2.x software release, a physical interface needed a dummy or valid IP address assigned to it before you could add the physical interface to a standby group. In the ACNS 5.3.1 software and later releases, this is no longer a requirement.
- Configure the duplex and speed settings of the member interfaces for better reliability.
- If the active interface fails, the operational interface in its standby group that is assigned the next highest priority becomes active. However, when the interface with the higher priority recovers, it does not become active again without manual intervention.
- If all the member interfaces of a standby group fail and then one recovers, the ACNS software brings up the standby group on the operational member interface.
- If a physical interface is a member of a port channel group, it cannot join a standby group. Likewise, if a physical interface is a member of a standby group, it cannot join a port channel group.

**Note**

Interface IP addresses and standby group IP addresses must be on different subnets to ensure reliable operation. Make sure to configure the interface default gateway using the `ip default-gateway` global configuration command instead of the `ip route` command.

To create standby groups on standalone Content Engines, use the `interface standby` global configuration command. In the ACNS 5.3.1 software release, the CLI syntax for configuring standby groups was changed to make it more similar to the port channel CLI syntax.

Table 16-2 lists the `interface standby` command options.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standby</td>
<td>Selects the standby group of interfaces to configure.</td>
</tr>
<tr>
<td>1</td>
<td>Specifies Standby Group 1.</td>
</tr>
<tr>
<td>2</td>
<td>Specifies Standby Group 2.</td>
</tr>
<tr>
<td>3</td>
<td>Specifies Standby Group 3.</td>
</tr>
<tr>
<td>4</td>
<td>Specifies Standby Group 4.</td>
</tr>
<tr>
<td>priority</td>
<td>Sets the priority of the member interface within a standby group. The priority of a member interface can be changed at runtime. The member interface that has the highest priority after this change becomes the new active interface (the default action is to preempt the currently active interface if an interface with higher priority exists).</td>
</tr>
</tbody>
</table>
Chapter 16      Configuring Additional Network Interfaces and Bandwidth on Standalone Content Engines

Configuring Additional Network Interfaces

### Note

Unlike port channels, standby groups do not support IP ACLs at a group level. However, you can configure a member interface of a standby group to support an IP ACL at the interface level. For example, you can individually configure the two member interfaces of Standby Group 1 (the Fast Ethernet slot 0/port 0 interface and the Fast Ethernet slot 0/port 1 interface) to support an IP ACL named ACL1 but you cannot configure the Standby Group 1 to support ACL1.

The following example shows how to create a standby group on a standalone Content Engine, and then add and remove members from this standby group:

**Step 1**  Create a standby group. In this case, the standby group is Standby Group 1.

```
ContentEngine# configure
ContentEngine(config)# interface standby 1
ContentEngine(config-if)#
```

**Step 2**  Assign a group IP address and netmask to Standby Group 1.

In ACNS 5.3.1 software and later releases, you can configure a group IP address regardless of whether the standby group is shut down or not.

In this example, Standby Group 1 is assigned a group IP address of 10.10.10.10 and a netmask of 255.0.0.0:

```
ContentEngine(config-if)# ip address 10.10.10.10 255.0.0.0
ContentEngine(config-if)# errors 500
```
Step 3

Define the member interfaces of Standby Group 1. Specify the priority of each member interface that is added to Standby Group 1.

The following example shows how to add two Fast Ethernet interfaces to Standby Group 1, and then assign each of these member interfaces a priority within the group:

a. A FastEthernet interface (slot 0/port 0) is added to Standby Group 1 and assigned a priority of 150.

```plaintext
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# standby 1 priority 150
```

b. A second FastEthernet interface (slot 0/port 1) is added to Standby Group 1 and assigned a priority of 100 (the default value).

```plaintext
ContentEngine(config)# interface FastEthernet 0/1
ContentEngine(config-if)# standby 1
ContentEngine(config-if)# exit
ContentEngine(config)#
```

Because FastEthernet 0/0 is assigned the highest priority (a priority number of 150) of all the member interfaces in the group, it will be chosen as the active interface for the group if it can be brought up.

Step 4

Specify the maximum number of transmit and receive errors that should be allowed on the active interface before the interface is shut down and the standby interface is brought up.

In this case, the maximum number of errors is set to 500.

```plaintext
ContentEngine(config-if)# errors 500
ContentEngine(config-if)# exit
ContentEngine(config)#
```

Step 5

Display information about the standby group configuration by entering the `show standby` EXEC command.

In the following sample command output, one standby group (Standby Group 1) is configured on this Content Engine. The command output also shows which member interface is the active interface. In this case, the active interface is the Fast Ethernet slot 0/port 0 interface.

```plaintext
CE-560# show standby
Standby Group: 1
   IP address: 10.10.10.10,   netmask: 255.255.255.0
   Member interface:
      FastEthernet 0/0   priority: 150
      FastEthernet 0/1   priority: 100
   Active interface: FastEthernet 0/0
CE-560#
```

*Note* To display information about a specific standby group configuration, enter the `show interface standby standby group number` EXEC command.

Step 6

To remove a member interface (for example, the Fast Ethernet slot 0/port 1 interface) from Standby Group 1, use the `no` form of the `standby` command:

```plaintext
ContentEngine(config)# interface FastEthernet 0/1
ContentEngine(config-if)# no standby 1
ContentEngine(config-if)# exit
ContentEngine(config)#
```
Step 7 To shut down a standby group, use the `interface standby` command to specify the group that you want to shut down and then enter the `shutdown` command to shut it down.

When a standby group is shut down, all of the alarms previously raised by this standby group are cleared.

The following example shows how to shut down Standby Group 1:

```
ContentEngine(config)# interface standby 1
ContentEngine(config-if)# exit
ContentEngine(config)# exit
```

Step 8 To tear down a standby group, use the `no` form of the `interface standby` command.

The following example shows how to tear down Standby Group 1:

```
ContentEngine(config)# interface standby 1
ContentEngine(config-if)# no ip address 10.10.10.10 255.0.0.0
Please remove member interface(s) from this standby group first.
ContentEngine(config)# interface GigabitEthernet 2/0
ContentEngine(config-if)# no standby 1
ContentEngine(config-if)# exit
ContentEngine(config)# interface standby 1
ContentEngine(config-if)# no ip address 10.10.10.10 255.0.0.0
ContentEngine(config-if)# exit
ContentEngine(config)# no interface standby 1
ContentEngine(config)# exit
```

## Specifying the Primary Interface

To specify the primary interface for the standalone Content Engine, use the `primary-interface` global configuration command.

```
primary-interface {FastEthernet 0-3/port | GigabitEthernet 1-2/port | PortChannel 1-2 | Standby group_num}
```

The following example shows how to specify the Fast Ethernet slot 0 port 0 as the primary interface on a Content Engine model CE-7320:

```
CE-7320(config)# primary-interface FastEthernet 0/0
```

To change the primary interface, reenter the command string and specify a different interface.

In the ACNS 5.2.1 software and later releases, you can select a standby interface as the primary interface (that is, you can enter the `primary-interface standby group-num` command to specify a standby group as the primary interface on a standalone Content Engine).

If you use the `restore factory-default preserve basic-config` command, the configuration for the primary interface is not preserved. On an ACNS 5.x device, if you want to reenable the ACNS network after using the `restore factory-default preserve basic-config` command, make sure to reconfigure the primary interface after the factory defaults are restored.
Configuring Multiple Secondary IP Addresses on a Single Physical Interface

You can assign up to four secondary IP addresses to a single physical interface on a Content Engine. By configuring multiple IP addresses on a single interface, the Content Engine can be present in more than one subnet. This allows you to optimize response time because the content goes directly from the Content Engine to the requesting client without being redirected through a router. The Content Engine becomes visible to the client because both are configured on the same subnet.

If a Content Engine has one physical interface that has multiple secondary IP addresses assigned to it, the egress traffic uses the source IP address that is chosen by IP routing. If the secondary IP addresses are in the same subnet as the primary IP address, then the egress traffic only uses the primary IP address. In contrast, if the secondary IP addresses are in a different subnet than the primary IP address, then the destination IP address determines which IP address on the Content Engine is used for the egress traffic.

To set these secondary IP addresses on a standalone Content Engine, use the `ip address` configuration interface command:

```
ContentEngine# configure
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# ip address 10.10.10.10 255.0.0.0 secondary
```

These secondary IP addresses become active only after the primary IP address is configured. No two interfaces can have the same IP address in the same subnet.

Configuring the Fibre Channel Interface

The ACNS 5.x software supports Fibre Channel interfaces. Fibre Channel is the chosen technology for interconnecting storage devices and servers in a storage area network (SAN). In a SAN, the storage does not need to be directly attached to the server, and data transfer happens over a high-throughput, high-availability network. Fibre Channel can operate at speeds of 1 gigabit per second (Gbps) and 2 Gbps.

To detect the presence of Fibre Channel storage, the storage array must be configured to assign storage space for the Content Engine, and the Content Engine must be reloaded before it can detect the storage assignment. To confirm whether the Content Engine has detected the storage assignment, use the `show disks` and the `show disks details` EXEC commands.

To configure the Fibre Channel interface on the Content Engine, use the `interface FibreChannel` interface configuration command. For example:

```
ContentEngine# configure
ContentEngine(config)# interface FibreChannel 0/0
ContentEngine(config-if)#?
   exit  Exit from this submode
   mode  Change the fibre channel interface operating mode
       no  Negate a command or set its defaults
   speed  Change the fibre channel interface speed
ContentEngine(config-if)# mode ?
    autosense  Use this mode to have the CE autosense
    direct-attached  Use this mode when the CE is directly connected to storage array
    switched  Use this mode when the CE is connected to a switch
ContentEngine(config-if)# speed ?
   1  1Gbps
   2  2Gbps
    autosense  autosense
```
Defining Interface Descriptions

In the ACNS 5.3.1 software and later releases, you can specify a one-line description for a specific interface on a Content Engine. You use the `description description text` interface configuration command to enter the description for the specific interface. The maximum length of the description text is 240 characters.

This feature is supported for the following interfaces: FastEthernet, GigabitEthernet, FibreChannel, PortChannel, and Standby. This example shows how to enter an interface description:

```
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# description "This an interface to the WAN."
```

Note

This feature is not currently supported for the SCSI or IDE interfaces.

After you define the description for an interface, you can use the `show EXEC` commands to display the defined interface descriptions. For example, to display all of the defined interface descriptions, enter this command:

```
ContentEngine# show running-config
```

interface FastEthernet 0/0
  description This is interface to WAN
  ip address dhcp
  ip address 192.168.1.200 255.255.255.0
  no autosense
  bandwidth 100
  full-duplex
  exit

To display the defined description for a specific interface on the Content Engine, enter the `show EXEC` interface type slot/port EXEC command. The following excerpt of the command output shows, the description of the specified interface is displayed as the first line in the command line output:

```
ContentEngine# show interface GigabitEthernet 1/0
DescriptionN This is the interface to the lab
```
Configuring Bandwidth for Interfaces and Content Services

With the various types of traffic originating from a device, every type of traffic, such as streaming media, HTTP, and metadata, consumes network resources.

Configuring Interface Bandwidth

To configure an interface bandwidth on a standalone Content Engine, use the `bandwidth` interface configuration command. Bandwidth is specified in megabits per second (Mbps). The 1000 Mbps option is not available on all ports and is the same as autosense.

```plaintext
bandwidth {10 | 100 | 1000}
no bandwidth {10 | 100 | 1000}
```

To restore default values, use the `no` form of this command.

For a Content Engine CE-7320 model that has an optical Gigabit Ethernet interface the speed of this interface cannot be changed. Therefore, Gigabit Ethernet interfaces only run at 1000 Mbps for a CE-7320. For newer models of the Content Engine (for example, the CE-510, CE-511, CE-565, CE-566, CE-7305, CE-7325, and CE-7326) that have a Gigabit Ethernet interface over copper, this restriction does not apply; you can configure these Gigabit Ethernet interfaces to run at 10, 100, or 1000 Mbps. On these newer Content Engine models, the 1000 Mbps setting implies autosense (for example, you cannot configure the Gigabit Ethernet interface to run at 1000 Mbps and half duplex). The ACNS 5.x software automatically enables autosense if the speed is set to 1000 Mbps.

In the ACNS 5.3.1 software and later releases, you can configure the Gigabit Ethernet interface settings (autosense, bandwidth, and duplex settings) if the Gigabit-over-copper interface is up or down. If the interface is up, it will apply the specific interface settings. If the interface is down, the specified settings are stored and then applied when the interface is brought up. For example, you can specify any of the following commands for a Gigabit-over-copper interface, which is currently down, and have these settings automatically applied when the interface is brought up:

```plaintext
ContentEngine(config-if)# bandwidth 10
ContentEngine(config-if)# bandwidth 100
ContentEngine(config-if)# bandwidth 1000
ContentEngine(config-if)# autosense
ContentEngine(config-if)# half-duplex
ContentEngine(config-if)# full-duplex
```

In the ACNS 5.2.x software and earlier releases, you could only configure the Gigabit Ethernet interface settings if the interface is up.

You cannot configure the Gigabit Ethernet interface settings if it is an optical Gigabit Ethernet interface (for example, if the Content Engine is a CE-7320 model).

With the ACNS 5.x software, you can also configure a maximum bandwidth for the preloading process using the `pre-load max-bandwidth` global configuration command.
Configuring Bandwidth for Content Services

To specify bandwidth limits for WMT and RealProxy live content or the streaming media that is being cached on the standalone Content Engine, use the `bandwidth` global configuration command.

For each type of content service (WMT and RealProxy), you can specify the maximum amount of bandwidth on the Content Engine that should be allocated to that service during a specified period. This is called scheduled bandwidth. For example, you can limit the RealProxy bandwidth to 1000 kbps from Monday at 8:00 a.m. to Friday at 6:00 p.m.

```
ContentEngine(config)# bandwidth allow 1000 real-proxy start-time monday 8:00 end-time friday 18:00
```

Although there are no default values for any of the bandwidth configuration fields, the values that you enter depend upon the bandwidth license in effect for your specific system. If you enter a value that is beyond the allowable bandwidth based on your system’s bandwidth capacity, the value is accepted but a warning message is displayed. Internally, your system bandwidth is limited to the maximum value granted by the license. All values entered are in kilobits per second (kbps).

For information about how to configure incoming and outgoing WMT bandwidth and bit rates on standalone Content Engines, see the “Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates” section on page 9-23.

In the ACNS 5.3.1 software and later releases, you can configure IP subnet-based bandwidth control for WMT requests. For more information on this topic, see the “Configuring Subnet-Based Outgoing Bandwidth” section on page 9-24.
Configuring Administrative Login Authentication and Authorization on Standalone Content Engines

This chapter describes how to configure administrative login authentication and authorization support for standalone Content Engines. This chapter describes how to configure a standalone Content Engine to use the local database and external RADIUS and TACACS+ databases to process log-in requests from administrators who want to access the Content Engine for configuration, monitoring, or troubleshooting purposes.

Note

Content authentication and authorization, which controls end users’ access to the requested content that is served through a standalone Content Engine, is independent of administrative login authentication and authorization for the Content Engine. For information about content authentication and authorization, see Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.”

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication. For information about configuring administrative login authentication and authorization for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

This chapter contains the following sections:

- Understanding Administrative Login Authentication and Authorization, page 17-2
- Configuring Administrative Login Authentication and Authorization, page 17-8
- Displaying the Current Administrative Authentication and Authorization Configuration, page 17-19
Understanding Administrative Login Authentication and Authorization

Administrative login authentication and authorization is used to control administrator access rights to the Content Engine. For example, if an administrator logs in to the Content Engine with the predefined ACNS software superuser account (root administrator), the Content Engine grants that administrator the highest privilege level (level 15), which allows that administrator to perform any Content Engine administrative task during that login session. For instance, that administrator could perform any of the following administrative tasks:

- Configure the Content Engine.
- Obtain statistical information that the Content Engine has collected.
- Reload the Content Engine.

For more information about managing administrative login accounts, see the “Managing Administrative Login Accounts” section on page 5-3.

Figure 17-1 shows how an administrator can log in to the Content Engine through the console or the Content Engine GUI. To process these administrative login requests, the Content Engine checks the specified authentication database to verify the user’s username and password and to determine the access rights that this particular administrator should be granted during this login session. When the Content Engine receives an administrative login request, the Content Engine can check its local database or a remote third-party database (the TACACS+ database or the RADIUS database) to verify the username with the password and to determine the access privileges of the administrator.

**Figure 17-1 Authentication Databases and a Standalone Content Engine**

![Diagram showing authentication databases and administrative login requests]

- Secure or non-secure access to Content Engine GUI
- Third-party AAA servers
- Console and Telnet sessions
- FTP
- SSH
- Content Engine local database (default primary authentication database)
Note

The ACNS 5.1 software and later releases support secure access or nonsecure access to the Content Engine GUI. (Either secure or nonsecure access to the Content Engine GUI is possible but not both simultaneously.)

The secured Content Engine GUI is the default (https://Content_Engine_ip_address:8003). For more information, see the “Logging in to the Content Engine GUI” section on page 4-55.

You can configure any combination of these authentication and authorization methods to control administrative login access to a standalone Content Engine:

- Local authentication and authorization—See the “Understanding Login Authentication and Authorization Through the Local Database” section on page 17-6.
- RADIUS—See the “Understanding RADIUS Authentication and Authorization” section on page 17-6.
- TACACS+—See the “Understanding TACACS+ Authentication and Authorization” section on page 17-7.

By default, the Content Engine uses the local login authentication method as the primary method to process administrative login requests. When you enable local authentication with one or more other authentication methods, local authentication is always attempted first if the priority flags (primary, secondary, or tertiary) are not set. You cannot specify different login authentication methods for console and Telnet connections.

Default Administrative Login Authentication and Authorization Configuration

By default, the Content Engine uses the local database to obtain login authentication and authorization privileges for administrative users.

Note

The authentication global configuration command configures the authentication methods that determine administrative login and configuration access to the Content Engine.

Table 17-1 lists the default configuration for administrative login authentication and authorization.

Table 17-1 Default Configuration for Administrative Login Authentication and Authorization

<table>
<thead>
<tr>
<th>Feature</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative login authentication</td>
<td>Enabled</td>
</tr>
<tr>
<td>Administrative configuration authorization</td>
<td>Enabled</td>
</tr>
<tr>
<td>Authentication server failover because the authentication server is unreachable</td>
<td>Disabled</td>
</tr>
<tr>
<td>TACACS+ login authentication (console and Telnet)</td>
<td>Disabled</td>
</tr>
<tr>
<td>TACACS+ authorization (console and Telnet)</td>
<td>Disabled</td>
</tr>
<tr>
<td>TACACS+ key</td>
<td>None specified</td>
</tr>
<tr>
<td>TACACS+ server timeout</td>
<td>5 seconds</td>
</tr>
<tr>
<td>TACACS+ retransmit attempts</td>
<td>2 times</td>
</tr>
</tbody>
</table>
You can change these defaults through the Content Engine CLI or GUI, as described in the “Configuring Administrative Login Authentication and Authorization” section on page 17-8.

### Understanding Failover for Administrative Login Authentication

By default, Content Engines fail over to the secondary method of administrative login authentication whenever the primary administrative login authentication method fails. In the ACNS software releases prior to the ACNS 5.0.5 software releases, you were not able to change this default method of failover for administrative login authentication.

In the ACNS 5.0.5 software and later releases, you can change this default login authentication failover method. For standalone Content Engines, you can use Content Engine GUI (choose System > Authentication and check the Failover due to Server Unreachable box) or the CLI (use the authentication fail-over server-unreachable global configuration command) to enable failover due to an unreachable server.

The following example sets failover for administrative login authentication to occur only if the authentication server is unreachable. In this case, the Content Engine will only query the next authentication method if the administrative login authentication server is unreachable.

```console
ContentEngine(config)# authentication fail-over server-unreachable
```

To use the login authentication failover feature, you must set TACACS+ or RADIUS as the primary login authentication method, and local as the secondary login authentication method.

If the failover due to unreachable server option is enabled, then remember the following information:

- Only two login authentication schemes (a primary and secondary scheme) are allowed on the Content Engine.
- The Content Engine will fail over from the primary authentication scheme to the secondary authentication scheme only if the specified authentication server is unreachable.
For example, if the failover due to the unreachable server option is enabled and RADIUS is set as the primary login authentication scheme and local is set as the secondary login authentication scheme, the following events occur:

- When the standalone Content Engine receives an administrative login request, it queries the RADIUS authentication server.
- If the RADIUS server is reachable, the standalone Content Engine uses this RADIUS database to authenticate the administrator.
- If the RADIUS server is not reachable, the standalone Content Engine tries the secondary authentication scheme (that is, it queries its local authentication database) to authenticate the administrator.

**Note**

Only if this RADIUS server is not reachable will the local database be contacted for authentication. In any other case (for example, if the authentication fails in the RADIUS server), then the local database is not contacted for authentication.

Conversely, if the failover due to unreachable server option is disabled, then the standalone Content Engine contacts the secondary authentication database regardless of the reason why the authentication failed with the primary authentication database.

If all the authentication databases are enabled for use, then all the databases are queried in the order of priority selected and based on the failover reason. If no failover reason is specified, then all the databases are queried in the order of their priority. For example, first the primary authentication database is queried, then the secondary authentication database is queried, and finally the tertiary database is queried.

The local and the remote databases (TACACS+ and RADIUS) can be enabled or disabled through the Content Engine CLI or GUI. The Content Engine verifies whether all databases are disabled and if so, sets the system to the default state (the local database is queried for authentication). (For information about this default state, see the “Default Administrative Login Authentication and Authorization Configuration” section on page 17-3.)

For more information on the various types of login authentication and authorization schemes, see the following sections:

- Understanding Login Authentication and Authorization Through the Local Database, page 17-6
- Understanding RADIUS Authentication and Authorization, page 17-6
- Understanding TACACS+ Authentication and Authorization, page 17-7

**Note**

For information about how to configure administrative login authentication and authorization on a standalone Content Engine, see the “Configuring Administrative Login Authentication and Authorization” section on page 17-8.
Understanding Administrative Login Authentication and Authorization Through the Local Database

Local authentication and authorization uses locally configured login and passwords to authenticate administrative login attempts. The login and passwords are local to each Content Engine and are not mapped to individual usernames.

By default, local login authentication is enabled first. You can disable local login authentication only after enabling one or more of the other administrative login authentication methods. However, when local login authentication is disabled, if you disable all other administrative login authentication methods, local login authentication is reenabled automatically.

Understanding RADIUS Authentication and Authorization

RADIUS is a client/server authentication and authorization access protocol used by a network access server (NAS) to authenticate users attempting to connect to a network device. The NAS functions as a client, passing user information to one or more RADIUS servers. The NAS permits or denies network access to a user based on the response it receives from one or more RADIUS servers. RADIUS uses the User Datagram Protocol (UDP) for transport between the RADIUS client and server.

You can configure a RADIUS key on the client and server. If you configure a key on the client, it must be the same as the one configured on the RADIUS servers. The RADIUS clients and servers use the key to encrypt all RADIUS packets transmitted. If you do not configure a RADIUS key, packets are not encrypted. The key itself is never transmitted over the network.

Note

For more information about how the RADIUS protocol operates, see the RFC 2138, *Remote Authentication Dial In User Service (RADIUS)*.

RADIUS authentication usually occurs in these instances:

- Administrative login authentication—When an administrator first logs in to the standalone Content Engine to configure the Content Engine for monitoring, configuration, or troubleshooting purposes. For more information, see the “Enabling and Disabling Administrative Login Authentication and Authorization Through RADIUS” section on page 17-17.
- HTTP request authentication—When an end user sends a service request that requires privileged access to content that is served by the Content Engine. For more information, see the “Configuring the RADIUS Authentication Service” section on page 10-19.

RADIUS authentication is disabled by default. You can enable RADIUS authentication and other authentication methods at the same time. You can also specify which method to use first. For more information about configuring RADIUS authentication, see the “Specifying RADIUS Authentication Settings for Standalone Content Engines” section on page 17-10.
Understanding TACACS+ Authentication and Authorization

TACACS+ controls access to network devices by exchanging Network Access Server (NAS) information between a network device and a centralized database to determine the identity of a user or an entity. TACACS+ is an enhanced version of TACACS, a UDP-based access-control protocol specified by RFC 1492. TACACS+ uses TCP to ensure reliable delivery and encrypt all traffic between the TACACS+ server and the TACACS+ daemon on a network device.

TACACS+ works with many authentication types, including fixed password, one-time password, and challenge-response authentication. TACACS+ authentication usually occurs in these instances:

- **Administrative login authentication**—When an administrator first logs in to the standalone Content Engine to configure the Content Engine for monitoring, configuration, or troubleshooting purposes. For more information, see the “Enabling and Disabling Administrative Login Authentication and Authorization Through TACACS+” section on page 17-18.

- **HTTP request authentication**—When an end user sends a service request that requires privileged access to content that is served by the Content Engine. For more information, see the “Configuring the TACACS+ Authentication Service” section on page 10-20.

When a user requests restricted services, TACACS+ encrypts the user password information using the MD5 encryption algorithm and adds a TACACS+ packet header. This header information identifies the packet type being sent (for example, an authentication packet), the packet sequence number, the encryption type used, and the total packet length. The TACACS+ protocol then forwards the packet to the TACACS+ server.

A TACACS+ server can provide authentication, authorization, and accounting functions. These services, while all part of TACACS+, are independent of one another, so a given TACACS+ configuration can use any or all of the three services.

When the TACACS+ server receives a packet, it does the following:

- Authenticates the user information and notifies the client that the login authentication has either succeeded or failed.

- Notifies the client that authentication will continue and that the client must provide additional information. This challenge-response process can continue through multiple iterations until login authentication either succeeds or fails.

You can configure a TACACS+ key on the client and server. If you configure a key on the Content Engine, it must be the same as the one configured on the TACACS+ servers. The TACACS+ clients and servers use the key to encrypt all TACACS+ packets transmitted. If you do not configure a TACACS+ key, packets are not encrypted.

TACACS+ authentication is disabled by default. You can enable TACACS+ authentication and local authentication at the same time.
TACACS+ Enable Password Attribute

The ACNS software CLI EXEC mode is used for setting, viewing, and testing system operations. It is divided into two access levels, user and privileged. To access privileged-level EXEC mode, enter the `enable` EXEC command at the user access level prompt and specify a privileged EXEC password (superuser or admin-equivalent password) when prompted for a password.

In TACACS+ there is an enable password feature that allows an administrator to define a different enable password per administrative-level user. If an administrative-level user logs in to the Content Engine with a normal-level user account (privilege level of 0) instead of an admin or admin-equivalent user account (privilege level of 15), that user must enter the admin password in order to access privileged-level EXEC mode. This requirement applies even if ACNS users are using TACACS+ for login authentication.

```
ContentEngine> enable
Password:
```

When using TACACS+ with ACNS, the maximum length for a password is 31 characters.

Configuring Administrative Login Authentication and Authorization

This section describes how to configure login authentication and authorization for ACNS administrators who want to log in to the Content Engine for monitoring, configuration, or troubleshooting purposes.

**Note**

Content authentication and authorization, which controls end users’ access to the requested content that is served through a standalone Content Engine, is independent of the administrative login authentication and authorization for the Content Engine.

For information about content authentication and authorization, see Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.”

To configure administrative login authentication and authorization for standalone Content Engines, follow these steps:

**Step 1**

Determine the login authentication scheme that you want to configure the standalone Content Engine to use when authenticating administrative login requests (for example, use the local database as the primary login database and your RADIUS server as the secondary authentication database).
Step 2 Configure the login authentication servers settings on the Content Engine (if a remote authentication database is to be used).

For example, specify the IP address of the remote RADIUS servers or TACACS+ servers that the Content Engine should use to authenticate login requests. For more information, see the following sections:

- Specifying RADIUS Authentication Settings for Standalone Content Engines, page 17-10
- Specifying TACACS+ Authentication Settings for Standalone Content Engines, page 17-12

Step 3 Specify the login authentication configuration scheme that the Content Engine should use to process administrative login requests:

- Specify the administrative login authentication scheme.
- Specify the administrative login authorization scheme.
- Specify the failover scheme for the administrative login authentication server (optional).

For example, specify which authentication database the Content Engine should check to process an administrative login request. For more information, see the “Specifying and Enabling the Administrative Login Authentication and Authorization Scheme” section on page 17-14.

Caution Make sure that RADIUS or TACACS+ authentication is configured and operating correctly before disabling local authentication and authorization. If you disable local authentication and RADIUS or TACACS+ is not configured correctly, or if the RADIUS or TACACS+ server is not online, you may be unable to log in to the Content Engine.

When local authentication is disabled, if you disable all other authentication methods, local authentication is reenabled automatically.

The following sections describe how to specify authentication server settings for a standalone Content Engine.

- Specifying RADIUS Authentication Settings for Standalone Content Engines, page 17-10
- Specifying TACACS+ Authentication Settings for Standalone Content Engines, page 17-12
Specifying RADIUS Authentication Settings for Standalone Content Engines

RADIUS authentication clients reside on the Content Engine running ACNS 5.x software. When enabled, these clients send authentication requests to a central (remote) RADIUS server, which contains login authentication and network service access information.

To configure RADIUS authentication on a standalone Content Engine, you must configure a set of RADIUS authentication server settings on the Content Engine. You can use the Content Engine GUI or the CLI to configure this set of RADIUS authentication server settings for a Content Engine.

Table 17-2 describes the RADIUS authentication settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADIUS server</td>
<td>RADIUS servers that the Content Engine is to use for RADIUS authentication. To enable the Content Engine to use a specific RADIUS server, enter the IP address or hostname of the RADIUS server and port information. Up to five different hosts are allowed. Early deployment of RADIUS was done using port number 1645, though the official port number for RADIUS is now 1812. Up to 5 different ports are allowed.</td>
</tr>
<tr>
<td>RADIUS key</td>
<td>Key used to encrypt and authenticate all communication between the RADIUS client (the standalone Content Engine) and the RADIUS server. The maximum number of characters in the key is 15. There is no default. Tip: Be sure the same RADIUS key is enabled on the RADIUS server.</td>
</tr>
<tr>
<td>RADIUS timeout interval</td>
<td>Number of seconds that the Content Engine waits for a response from the specified RADIUS authentication server before declaring a timeout. The range is 1 to 20 seconds. The default value is 5 seconds.</td>
</tr>
<tr>
<td>RADIUS retransmit count</td>
<td>Number of times the Content Engine is to retransmit its connection to the RADIUS if the RADIUS timeout interval is exceeded. The range is 1 to 3 tries. The default value is 2 tries.</td>
</tr>
</tbody>
</table>

After configuring these RADIUS authentication settings on the Content Engine, you can enable the following types of RADIUS authentication on the Content Engine:

- RADIUS login authentication and authorization, as described in the “Enabling and Disabling Administrative Login Authentication and Authorization Through RADIUS” section on page 17-17.
- RADIUS HTTP request authentication, as described in the “Configuring the RADIUS Authentication Service” section on page 10-19.

To use the Content Engine GUI to configure RADIUS authentication settings on a standalone Content Engine, choose Caching > RADIUS. Use the displayed RADIUS Authentication Settings window. Click the Enable RADIUS On radio button to enable RADIUS authentication on this Content Engine. Use the RADIUS Authentication Settings window to specify the other RADIUS authentication settings. For more information about this window, click the HELP button in the window.
To use the Content Engine CLI to configure RADIUS authentication settings on a standalone Content Engine, follow these steps:

**Step 1** Specify one or more RADIUS servers. Optionally, specify the destination UDP port to use on the server. The default port is 1645.

```
ContentEngine(config)# radius-server host ip_addr [auth-port port]
```

This example shows how to specify a RADIUS server at 172.16.52.3:

```
ContentEngine(config)# radius-server 172.16.52.3
```

**Step 2** Specify the RADIUS key on the Content Engine.

```
ContentEngine(config)# radius-server key myradiuskey
```

**Step 3** Specify the RADIUS timeout interval.

For example, configure the Content Engine to wait 10 seconds before declaring a timeout if it has not received a response from the RADIUS server:

```
ContentEngine(config)# radius-server timeout 10
```

**Step 4** Specify the RADIUS retransmit count.

For example, configure the Content Engine to retransmit three times to the RADIUS server if a RADIUS timeout occurs:

```
ContentEngine(config)# radius-server retransmit 3
```

**Note** For more information about a RADIUS authentication setting (for example, a RADIUS key), see Table 17-2. For more detailed information about the `radius-server` global configuration command, see the *Cisco ACNS Software Command Reference, Release 5.5* publication.

The following example enables the RADIUS client on the Content Engine, specifies a remote RADIUS server for authentication, specifies the RADIUS key on the Content Engine, accepts retransmit defaults, and excludes the domain name and mydomain.net domains from RADIUS authentication. Configuration can be verified with the `show radius-server` and `show rule all` EXEC commands.

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

You can now enable RADIUS as an administrative login authentication and authorization method for this Content Engine, as described in the “Enabling and Disabling Administrative Login Authentication and Authorization Through RADIUS” section on page 17-17.
Specifying TACACS+ Authentication Settings for Standalone Content Engines

In order to configure TACACS+ authentication on standalone Content Engines, you must configure a set of TACACS+ authentication settings on the Content Engine. You can use the Content Engine CLI or GUI to configure this set of TACACS+ authentication settings for a standalone Content Engine.

Table 17-3 describes the TACACS+ authentication settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TACACS+ server</td>
<td>TACACS+ servers that the Content Engine is to use for TACACS+ authentication. Explicitly specify the primary TACACS+ server; otherwise, the Content Engine makes its own decision. You can configure one primary TACACS+ server and two backup TACACS+ servers. TACACS+ uses the standard port (port 49) for communication, based on the specified service.</td>
</tr>
<tr>
<td>TACACS+ key</td>
<td>Secret key that the Content Engine will use to communicate with the TACACS+ server. The maximum number of characters in the TACACS+ 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. There is no default. Tip Be sure the same TACACS+ key is specified on the TACACS+ server.</td>
</tr>
<tr>
<td>TACACS+ timeout interval</td>
<td>Number of seconds that the Content Engine will wait for a response from the specified TACACS+ authentication server before declaring a timeout. The range is 1 to 20 seconds. The default value is 5 seconds.</td>
</tr>
<tr>
<td>TACACS+ retransmit count</td>
<td>Number of times that the Content Engine is to retransmit its connection to the TACACS+ if the TACACS+ timeout interval is exceeded. The range is 1 to 3 tries. The default value is 2 tries.</td>
</tr>
<tr>
<td>TACACS+ password</td>
<td>Method for password authentication. By default, the Password Authentication Protocol (PAP) is used for password authentication. The other option is to use ASCII clear text.</td>
</tr>
</tbody>
</table>

To use the Content Engine CLI to configure TACACS+ authentication settings on a standalone Content Engine, follow these steps:

**Step 1** Specify one or more TACACS+ servers.

```
ContentEngine(config)# tacacs server ip_addr [primary]
```

This example shows how to specify a specific TACACS+ server as a primary server:

```
ContentEngine(config)# tacacs server 172.16.50.1 primary
```

This example shows how to specify a specific TACACS+ server as a backup server. This can be achieved by not specifying the `primary` option:

```
ContentEngine(config)# tacacs server 172.16.50.2
```

**Step 2** Specify the TACACS+ key.

```
ContentEngine(config)# tacacs key key
```
Step 3 Specify the TACACS+ timeout interval.
For example, configure the Content Engine to wait 15 seconds before declaring a timeout if it has not received a response from the TACACS+ server:

ContentEngine(config)# tacacs timeout 15

Step 4 Specify the TACACS+ retransmit count.
For example, configure the Content Engine to retransmit only one time to the TACACS+ server if a TACACS+ timeout occurs:

ContentEngine(config)# tacacs retransmit 1

Step 5 Specify the method for TACACS+ password authentication.
For example, specify ASCII clear text by entering the ASCII keyword:

ContentEngine(config)# tacacs password ascii

Note
For more information about a TACACS+ authentication setting (for example, specifying a TACACS+ key), see Table 17-3. For more detailed information about the tacacs server global configuration command, see the Cisco ACNS Software Command Reference, Release 5.5 publication.

In the following example, a TACACS+ server with the hostname of spearhead is configured as the primary TACACS+ server. The Content Engine is configured to use the same key (human789) as the one used on the TACACS+ server (the server named spearhead) and the default timeout interval, number of retransmits, and password types are changed. This example also shows how to use the show tacacs EXEC command to view the current TACACS+ configuration on the Content Engine:

ContentEngine(config)# tacacs host spearhead primary
ContentEngine(config)# tacacs key human789
ContentEngine(config)# tacacs timeout 10
ContentEngine(config)# tacacs retransmit 5
ContentEngine(config)# tacacs password ascii
ContentEngine(config)# exit

ContentEngine# show tacacs
Login Authentication for Console/Telnet Session: enabled (secondary)
Configuration Authentication for Console/Telnet Session: enabled (secondary)

TACACS+ Configuration:
-----------------------
TACACS+ Authentication is off
Key       = *****
Timeout   = 5
Retransmit = 2
Password type: ascii

Server     Status
------------ ------
10.107.192.148 primary
10.107.192.168
10.77.140.77

You can now enable TACACS+ as an administrative login authentication and authorization method for this Content Engine, as described in the “Enabling and Disabling Administrative Login Authentication and Authorization Through TACACS+” section on page 17-18.
Specifying and Enabling the Administrative Login Authentication and Authorization Scheme

This section describes how to define and modify the various administrative login authentication and authorization schemes (the authentication configuration) for a standalone Content Engine:

- **Usage Guidelines, page 17-14**
- **Reenabling and Disabling Administrative Login Authentication and Authorization Through the Local Database, page 17-16**
- **Enabling and Disabling Administrative Login Authentication and Authorization Through RADIUS, page 17-17**
- **Enabling and Disabling Administrative Login Authentication and Authorization Through TACACS+, page 17-18**

**Caution**

Make sure that RADIUS or TACACS+ authentication is configured and operating correctly before disabling local authentication and authorization. If you disable local authentication and if RADIUS or TACACS+ is not configured correctly, or if the RADIUS or TACACS+ server is not online, you may be unable to log in to the Content Engine.

**Usage Guidelines**

When defining or modifying the authentication configuration method for a standalone Content Engine, remember the following important information:

- You can use the Content Engine GUI or the CLI to choose between using an external access server or the internal (local) Content Engine-based AAA system for user access management.
- You can configure any combination of these authentication and authorization methods to control access and set privileges on a standalone Content Engine:
  - Local authentication and authorization
  - RADIUS authentication and authorization
  - TACACS+ authentication and authorization
- To configure the administrative login authentication and authorization (configuration) options, use the `authentication` global configuration command:

  ```
  authentication {configuration [local | radius | tacacs] enable [primary | secondary | tertiary] | fail-over server-unreachable | login [local | radius | tacacs] enable [primary | secondary | tertiary]}
  ```

  Table 17-4 describes the parameters for the `authentication` global configuration command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration</td>
<td>Sets configuration authentication (authorization).</td>
</tr>
<tr>
<td>local</td>
<td>Selects the local method for authentication.</td>
</tr>
<tr>
<td>radius</td>
<td>Selects the RADIUS server for authentication.</td>
</tr>
<tr>
<td>tacacs</td>
<td>Selects the TACACS+ server for authentication.</td>
</tr>
</tbody>
</table>
The authentication global configuration command configures both administrative login and configuration access to the standalone Content Engine.

The authentication login local and authentication configuration local global configuration commands use a local database for authentication and authorization:

- The authentication login command specifies the administrative login authentication method used to determine whether the administrator has any level of permission to access the Content Engine.
- The authentication configuration command determines the privileges (level of user access to the Content Engine) for authenticated administrators.

The authentication login radius and authentication configuration radius global configuration commands use a remote RADIUS server to determine the level of administrative access.

By default, the local method is enabled, with TACACS+ and RADIUS both disabled for administrative login and configuration. Whenever TACACS+ and RADIUS are disabled, the local method is automatically enabled. TACACS+, RADIUS, and local methods can be enabled at the same time.

- The primary option specifies the first method to attempt for both administrative login and configuration.
- The secondary option specifies the method to use if the primary method fails.
- The tertiary option specifies the method to use if both the primary and the secondary methods fail.

If all methods of an authentication login or authentication configuration command are configured as primary, or all as secondary or tertiary, the local method is attempted first, then TACACS+, and then RADIUS.

The following example enables local, TACACS+, and RADIUS authentication and authorization, setting TACACS+ as the first method used, local as the secondary method if the TACACS+ method fails, and RADIUS as the tertiary method to use if both local and TACACS+ fail:

```
ContentEngine(config)# authentication login tacacs enable primary
ContentEngine(config)# authentication login local enable secondary
ContentEngine(config)# authentication login radius enable tertiary
ContentEngine(config)# authentication configuration tacacs enable primary
ContentEngine(config)# authentication configuration local enable secondary
ContentEngine(config)# authentication configuration radius enable tertiary
```
Chapter 17      Configuring Administrative Login Authentication and Authorization on Standalone Content Engines

Configuring Administrative Login Authentication and Authorization

Note

The **tacacs** global configuration command and a TACACS+ server must be configured to use the TACACS+ authentication and authorization method. For information about configuring a TACACS+ server, see the “Specifying TACACS+ Authentication Settings for Standalone Content Engines” section on page 17-12.

The **radius-server** global configuration command and a RADIUS server must be configured to use the RADIUS authentication and authorization method. For information about configuring a RADIUS server, see the “Specifying RADIUS Authentication Settings for Standalone Content Engines” section on page 17-10.

- Authentication configuration applies to the following:
  - Console and Telnet connection attempts
  - Secure FTP (SFTP), SSH (SSH Version 1 and Version 2), and Websense server access
- If you configure a RADIUS or TACACS+ key on the Content Engine (the RADIUS and the TACACS+ client), make sure that you configure an identical key on the RADIUS or TACACS+ server.
- If you configure multiple RADIUS or TACACS+ servers, the first server configured is the primary server, and authentication requests are sent to this server first. You can also specify secondary and tertiary servers for authentication and authorization purposes.
  - You can specify a server as primary, secondary, or tertiary by using the **primary**, **secondary**, or **tertiary** keywords in the **authentication** global configuration command.
  - You can also specify a server as primary, secondary, or tertiary from the Content Engine GUI. Choose **System > Authentication** and then choose **Primary**, **Secondary**, or **Tertiary** from the drop-down list next to the appropriate server.
- By default, the Content Engine uses the local database to authenticate and authorize administrative login requests. The Content Engine verifies whether all authentication databases are disabled and if so, sets the system to the default state. For information on this default state, see the “Default Administrative Login Authentication and Authorization Configuration” section on page 17-3.

Reenabling and Disabling Administrative Login Authentication and Authorization Through the Local Database

By default, the Content Engine is configured to use its local database to authenticate and authorize administrative login requests. This scheme of authentication and authorization is referred to as the local method. You can use the Content Engine GUI or CLI to disable and reenable this method of authentication and authorization on a standalone Content Engine.

Caution

Make sure that RADIUS or TACACS+ authentication is configured and operating correctly before disabling local authentication and authorization. If you disable local administrative authentication and if RADIUS or TACACS+ is not configured correctly, or if the RADIUS or TACACS+ server is not online, you may be unable to log in to the Content Engine.
Configuring Administrative Login Authentication and Authorization on Standalone Content Engines

Chapter 17      Configuring Administrative Login Authentication and Authorization on Standalone Content Engines

Configuring Administrative Login Authentication and Authorization

If you have disabled it on a Content Engine and want to reenable it from the Content Engine GUI, choose **System > Authentication**. In the displayed Authentication Configuration window, check the **Enable** box next to Local to enable local login authentication. By default, the local database is the primary database for administrative login authentication. To change this default, choose another option (for example, **Secondary** or **Tertiary**) from the drop-down list next to **Local**. Click **Update**. For more detailed information about how to use the Authentication Configuration window to perform this task, click the **HELP** button in the window.

To use the Content Engine CLI to reenable the local method on a standalone Content Engine, follow these steps:

**Step 1**  
Reenable local login authentication.  
ContentEngine(config)# **authentication login local enable**

**Step 2**  
Reenable local authorization of administrative users (control their privileges during the session).  
ContentEngine(config)# **authentication configuration local enable**

There are two privilege levels that can be granted to administrative users: normal-level administrative access (restricted privilege level of 0, or superuser administrative access [privilege level of 15]). For more information about privilege levels for administrative users, see the “Managing Administrative Login Accounts” section on page 5-3.

---

**Note**  
To disable local administrative authentication and authorization on a standalone Content Engine, use the **no** form of the **authentication** global configuration command (for example, use the **no authentication login local enable** command to disable local administrative authentication).

---

**Enabling and Disabling Administrative Login Authentication and Authorization Through RADIUS**

When configuring a standalone Content Engine to use RADIUS to authenticate and authorize administrative login requests, keep these important points in mind:

- By default, RADIUS authentication and authorization is disabled on a standalone Content Engine.
- Before enabling RADIUS authentication on the Content Engine, you must specify at least one RADIUS server for the Content Engine to use. For information about specifying a RADIUS server, see the “Specifying RADIUS Authentication Settings for Standalone Content Engines” section on page 17-10.
- You can enable RADIUS authentication and other authentication methods at the same time. You can specify which method to use first using the **primary** keyword. When local authentication is disabled, if you disable all other authentication methods, local authentication is reenabled automatically.
- You can use the Content Engine GUI or the CLI to enable RADIUS authentication and authorization on a standalone Content Engine.

From the Content Engine GUI, choose **System > Authentication**. Use the displayed Authentication Configuration window. For more information about how to use the Authentication Configuration window, click the **HELP** button in the window.
To use the Content Engine CLI to enable RADIUS authentication and authorization on a standalone Content Engine, follow these steps:

**Step 1** Enable RADIUS authentication for normal login mode.

```
ContentEngine(config)# authentication login radius enable [primary] [secondary] [tertiary]
```

For example, to force the Content Engine to try RADIUS authentication first (to try it before using TACACS+ authentication), enter the following command:

```
ContentEngine(config)# authentication login radius enable primary
```

**Step 2** Enable RADIUS authorization.

```
ContentEngine(config)# authentication configuration radius enable [primary] [secondary] [tertiary]
```

For example, to force the Content Engine to try RADIUS authorization first (to try it before using TACACS+ authorization), enter the following command:

```
ContentEngine(config)# authentication configuration radius enable primary
```

**Note**
To disable RADIUS authentication and authorization on a standalone Content Engine, use the `no` form of the `authentication` global configuration command (for example, use the `no authentication login radius enable` command to disable RADIUS authentication).

### Enabling and Disabling Administrative Login Authentication and Authorization Through TACACS+

When configuring a standalone Content Engine to use TACACS+ to authenticate and authorize administrative login requests, keep this important points in mind:

- By default, TACACS+ authentication and authorization is disabled on a standalone Content Engine.
- The `authentication login tacacs` and `authentication configuration tacacs` commands use a remote TACACS+ server for administrative login authentication and authorization, and to determine the level of administrative access.
- Before enabling TACACS+ authentication on the Content Engine, you must specify at least one TACACS+ server for the Content Engine to use. For information on specifying a TACACS+ server, see the “Specifying TACACS+ Authentication Settings for Standalone Content Engines” section on page 17-12.
- If you are using both RADIUS and TACACS+, you can use the `primary` keyword to force the Content Engine to try TACACS+ authentication first.
- You can use the Content Engine GUI or the CLI to enable TACACS+ authentication and authorization on a standalone Content Engine.

To enable TACACS+ authentication and authorization from the Content Engine GUI, choose `System > Authentication`, and use the displayed Authentication Configuration window. For more information about how to use the Authentication Configuration window, click the HELP button in the window.
To use the Content Engine CLI to enable TACACS+ authentication and authorization on a standalone Content Engine, follow these steps:

---

**Step 1**
Enable TACACS+ authentication for normal login mode.

```
ContentEngine(config)# authentication login tacacs enable [primary] [secondary] [tertiary]
```

For example, to force the Content Engine to try TACACS+ authentication first (to try it before using RADIUS authentication), enter this command:

```
ContentEngine(config)# authentication login tacacs enable primary
```

**Step 2**
Enable TACACS+ authorization.

```
ContentEngine(config)# authentication configuration tacacs enable [primary] [secondary] [tertiary]
```

For example, to force the Content Engine to try TACACS+ authorization first (to try it before using RADIUS authorization), enter this command:

```
ContentEngine(config)# authentication configuration tacacs enable primary
```

---

**Note**
To disable TACACS+ authentication and authorization on a standalone Content Engine, use the `no` form of the `authentication` global configuration command (for example, use the `no authentication login tacacs enable` global configuration command to disable TACACS+ authentication).

---

**Displaying the Current Administrative Authentication and Authorization Configuration**

To display the current administrative login authentication and authentication configuration on a standalone Content Engine, enter the `show authentication user` EXEC command. As the following sample output shows, the authentication schemes (for example, local, RADIUS, or TACACS+) that the Content Engine is configured to use to authenticate and authorize administrative login requests are displayed:

```
ContentEngine# show authentication user
Authentication scheme fail-over reason: server unreachable

Login Authentication: Console/Telnet Session
----------------------------- -----------------------
local                         enabled (primary)
radius                        disabled
tacacs                        disabled

Configuration Authentication: Console/Telnet Session
----------------------------- -----------------------
local                         enabled (primary)
radius                        disabled
tacacs                        disabled
```

---
Collectively, authentication, authorization, and accounting are often referred to as AAA. AAA accounting is the action of keeping track of administrative user activities for system accounting purposes.

This chapter describes how to configure AAA accounting for standalone Content Engines that are running the ACNS 5.4.x software and later releases. This chapter contains the following sections:

- About AAA Accounting, page 18-2
- Configuring AAA Accounting Settings on Standalone Content Engines, page 18-5
- Displaying the AAA Accounting Configuration for Standalone Content Engines, page 18-8
- Displaying and Clearing AAA Accounting Statistics on Standalone Content Engines, page 18-8

Transaction monitoring (recording information about end users’ content requests) is tracked in the ACNS software transaction logs. Transaction monitoring is independent of AAA accounting that tracks administrative user activities. For more information on transaction monitoring, see the “Monitoring Transactions with Standalone Content Engines” section on page 21-27.

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication. For information about configuring AAA accounting for Content Engines that are registered with a Content Distribution Manager, see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
About AAA Accounting

AAA accounting tracks the activities of an administrative user, and can be used for system accounting purposes (for example, as an audit trail, basis for billing for connection time or the resources used [bytes transferred], reporting purposes, or security purposes). In the ACNS 5.2 software and later releases, AAA accounting is supported with TACACS+; RADIUS is not currently supported.

The TACACS+ protocol allows effective communication of AAA information between Content Engines and a central server. TACACS+ uses TCP for reliable connections between clients and servers. Content Engines send authentication and authorization requests, as well as accounting information, to the specified TACACS+ server. By configuring AAA accounting with TACACS+, you can store this AAA information in a central database.

You can activate AAA accounting for four different types of events. (See Table 18-1.)

<table>
<thead>
<tr>
<th>Type of Event</th>
<th>Description and More Information</th>
<th>Corresponding Content Engine CLI Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>System event</td>
<td>System accounting for all system-level events that are not associated with administrators, such as reloads. See the “About System Accounting” section on page 18-3.</td>
<td>aaa accounting system default { start-stop</td>
</tr>
<tr>
<td>Exec shell and login/logout events</td>
<td>EXEC shell accounting for EXEC processes (user shells). See the About EXEC Shell Accounting, page 18-3 for more information.</td>
<td>aaa accounting exec default { start-stop</td>
</tr>
<tr>
<td>Normal (nonsuperuser) administrative CLI commands</td>
<td>Command accounting for all CLI commands that are executed on the Content Engine by an administrator who has normal privileges. See the “About Command Accounting” section on page 18-4 for more information.</td>
<td>aaa accounting commands 0 default { start-stop</td>
</tr>
<tr>
<td>Superuser administrative CLI commands</td>
<td>Command accounting for all CLI commands that are executed on the Content Engine by a superuser. See the “About Command Accounting” section on page 18-4 for more information.</td>
<td>aaa accounting commands 15 default { start-stop</td>
</tr>
</tbody>
</table>

AAA accounting and transaction monitoring are independent of each other:

- With administrative login requests, an administrator is logging in to the Content Engine for configuration, monitoring, or troubleshooting purposes. The administrator is using the predefined superuser administrative account, or another administrative account that has been created on the Content Engine. The Content Engine processes the administrative login request using one or more of the following login authentication methods: the local database, an external RADIUS server, or an external TACACS+ server. This chapter describes how to configure AAA accounting for administrative login requests and activities. For information about how to configure AAA authentication and authorization (administrative login authentication and authorization on the Content Engine), see Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines.”

- With content requests, end users (web clients) are using their browsers or media players on their desktops to request content that is served through the Content Engine. The Content Engine tracks end users’ access to content that is served through the Content Engine, and records information (for example, which user is accessing what content and for how long) about these content requests in the ACNS software transaction logs. For information about configuring content authentication and
About AAA Accounting

Authorization, see Chapter 10, “Configuring Content Authentication and Authorization on Standalone Content Engines.” For information about the ACNS software transaction logs, see the “Monitoring Transactions with Standalone Content Engines” section on page 21-27.

About System Accounting

System accounting provides information about all system-level events (for example, a system reboot). You can access system accounting information through the TACACS+ server’s accounting log file. This log file uses the following report format for this type of accounting information:

WeekDay#Month#Day#Time#Year#CEaddress#username#terminal#RemoteHost#Event#EventTime#TaskId#Timezone#SystemService#SystemAccountingEvent#EventReason

The following are some examples of the system accounting report that is available on the TACACS+ server:

Wed Apr 14 08:37:14 2004 172.16.0.0 unknown unknown 0.0.0.0 start start_time=1081909831
task_id=2725 timezone=PST service=system event=sys_acct reason=reload
Wed Apr 14 10:19:18 2004 172.16.0.0 admin ttyS0 0.0.0.0 stop stop_time=1081915955
task_id=5358 timezone=PST service=system event=sys_acct reason=shutdown

About EXEC Shell Accounting

EXEC shell accounting is used to report the events of an administrator logging in and out of the EXEC shell through Telnet, FTP, or SSH (SSH Version 1 or Version 2). This type of accounting provides information about user EXEC terminal sessions (user shells) events, including username, date, start and stop times, and the IP address of the accessed server (for example, the IP address of the FTP server). The EXEC shell accounting information can be accessed through the TACACS+ server’s accounting log file. This log file uses the following report format for this type of accounting information:

WeekDay#Month#Day#Time#Year#CEaddress#username#terminal#RemoteHost#Event#EventTime#TaskId#Timezone#Service

The following are some examples of the EXEC shell accounting report that is available on the TACACS+ server:

Wed Apr 14 11:19:19 2004 172.16.0.0 super10 pts/0 172.31.0.0 start start_time=1081919558
task_id=3028 timezone=PST service=shell
Wed Apr 14 11:19:23 2004 172.16.0.0 super10 pts/0 172.31.0.0 stop stop_time=1081919562
task_id=3028 timezone=PST service=shell
Wed Apr 14 11:22:13 2004 172.16.0.0 normal20 pts/0 via5.abc.com start
start_time=1081919732 task_id=3048 timezone=PST service=shell
Wed Apr 14 11:22:16 2004 172.16.0.0 normal20 pts/0 via5.abc.com stop
stop_time=1081919775 task_id=3048 timezone=PST service=shell
Wed Apr 14 11:25:29 2004 172.16.0.0 admin ftp via5.abc.com start start_time=1081919928
task_id=3069 timezone=PST service=shell
Wed Apr 14 11:25:33 2004 172.16.0.0 admin ftp via5.abc.com stop stop_time=1081919931
task_id=3069 timezone=PST service=shell
About AAA Accounting

The Content Engine records information about each CLI command that is executed on the Content Engine (whether in EXEC mode or configuration mode) that is executed. The accounting record for each command includes the following information:

- The syntax of the executed command.
- The username of the administrator who executed the particular CLI command.
- The privilege level of the administrator who executed the particular CLI command.

Normal privileges (privilege level of 0) allows restricted access to the Content Engine, and superuser privileges (privilege level of 15) allows unrestricted access to the Content Engine for monitoring, configuration, or troubleshooting purposes. The command account reports the same privilege level for all configuration and EXEC mode CLI commands that are executed by a particular administrator.

The recorded privilege level of the CLI commands is the same as the logged-in user’s privilege level:

- Administrators with superuser privileges will log a privilege level of 15 in the accounting record.
- Administrators with normal privileges will log a privilege level of 0 in the accounting record.

- The date and time that each CLI command was executed.

The command accounting information can be accessed through the TACACS+ server’s accounting log file. This log file uses the following report format for this type of accounting information:

WeekDay#Month#Day#Time#Year#CEaddress#username#terminal#RemoteHost#Event#EventTime#TaskId#Timezone#Service#PrivilegeLevel#CLICommand

The following are some examples of the command accounting report that is available on the TACACS+ server:

Wed Apr 14 12:35:38 2004 172.16.0.0 admin ttyS0 0.0.0.0 start start_time=1081924137
task_id=3511 timezone=PST service=shell -lvl=0 cmd=logging console enable
Wed Apr 14 12:35:39 2004 172.16.0.0 admin ttyS0 0.0.0.0 stop stop_time=1081924137
task_id=3511 timezone=PST service=shell priv-lvl=0 cmd=/logging console enable

In addition to command accounting, the Content Engine records any executed CLI command in the system log (syslog). The message format is as follows:

ce_syslog(LOG_INFO, CESM_PARSER, PARSER_ALL, CESM_350232, "CLI_LOG $s: %s \n", __FUNCTION__, pd->command_line);
Configuring AAA Accounting Settings on Standalone Content Engines

When configuring AAA accounting on a standalone Content Engine, keep these important points in mind:

- The Content Engine sends AAA accounting information only to the TACACS+ server, and does not send it to the console or to any other device.
- AAA accounting through RADIUS is not currently supported.
- By default, AAA accounting is disabled on a Content Engine. To enable and configure AAA accounting on a standalone Content Engine, you must use the Content Engine CLI. (Currently, this feature cannot be configured through the Content Engine GUI.)
- When activating AAA accounting, there are three command options to indicate when TACACS+ accounting is to occur: stop-only, start-only, and wait-start.

**Caution**

The ACNS software displays the following warning message if the wait-start option is configured.

Warning: The device may become non-responsive if it cannot contact a configured TACACS+ server.

The administrator is asked to confirm the configuration in an indefinite loop until the administrator enters yes" to the following prompt:

Are you sure you want to proceed? [yes]

Table 18-2 describes the stop-only, start-only, and wait-start options.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Content Engine CLI Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>start-stop</td>
<td>The Content Engine sends the TACACS+ accounting server a start record accounting notice at the beginning of a process and a stop record at the end of the process. The start accounting record is sent in the background. The requested user process begins regardless of whether or not the start accounting record was acknowledged by the TACACS+ accounting server.</td>
<td>aaa accounting { commands</td>
</tr>
<tr>
<td>stop-only</td>
<td>The Content Engine sends a stop record accounting notice at the end of the specified activity or event to the TACACS+ accounting server.</td>
<td>aaa accounting { commands</td>
</tr>
<tr>
<td>wait-start</td>
<td>The Content Engine sends both a start and a stop accounting record to the TACACS+ accounting server. However, the requested user service does not begin until the TACACS+ accounting server acknowledges the start accounting record. A stop accounting record is also sent.</td>
<td>aaa accounting { commands</td>
</tr>
</tbody>
</table>
Configuring AAA Accounting Settings on Standalone Content Engines

To configure a standalone Content Engine to use TACACS+ to support AAA accounting, follow these steps:

---

**Step 1**

Make sure that at least one TACACS+ server is configured for the standalone Content Engine.

Before you can configure the AAA accounting settings for a standalone Content Engine, you must first configure a TACACS+ server for the Content Engine. For example, you must specify the TACACS+ key and hostname or IP address of the TACACS+ server to which the Content Engine will send its AAA information. The Content Engines does not have a predefined TACACS+ server configuration.

*a.* Specify the TACACS+ key on the Content Engine.

```
ContentEngine(config)# tacacs key key
```

$key$ is the secret key that the Content Engine will use to communicate with the TACACS+ server. There is no default. Be sure the same TACACS+ key is also specified on the TACACS+ server.

For example, to specify abc as the key, enter:

```
ContentEngine(config)# tacacs key abc
```

*b.* Specify a specific TACACS+ server as the accounting server.

Explicitly specify the primary TACACS+ server; otherwise, the Content Engine makes its own decision. You can configure one primary TACACS+ server and two backup TACACS+ servers. TACACS+ uses port 49 as a standard port for communication.

To specify one or more TACACS+ servers, enter:

```
ContentEngine(config)# tacacs server ip_addr [primary]
```

In this example, the TACACS+ server with the IP address 172.16.50.1 is explicitly configured as the primary server through the use of the $primary$ option:

```
ContentEngine(config)# tacacs server 172.16.50.1 primary
```

In this example, the TACACS+ server with the IP address 172.16.50.2 is configured as a backup server. This server is configured as a backup server because the $primary$ keyword is not specified.

```
ContentEngine(config)# tacacs server 172.16.50.2
```

For more information about configuring a TACACS+ server for standalone Content Engines, see the “Specifying TACACS+ Authentication Settings for Standalone Content Engines” section on page 17-12.

**Step 2**

Activate accounting for system events and indicate when accounting is to take place using the `aaa accounting system default {start-stop | stop-only | wait-start} tacacs` global configuration command.

---

Note: See Table 18-2 for a description of the $start-stop$, $stop-only$, and $wait-start$ options of the `aaa accounting` global configuration commands.

In this example, the Content Engine is configured to record all system activities. The command also configures the Content Engine to send the TACACS+ server a stop record accounting notice at the end of the specified activity or event.

```
ContentEngine(config)# aaa accounting system default stop-only tacacs
```
Step 3  Activate accounting for EXEC mode processes and indicate when accounting is to take place by using the `aaa accounting exec default {start-stop | stop-only | wait-start} tacacs` global configuration command.

In this example, the Content Engine is configured to record all user EXEC sessions. The command also configures the Content Engine to send the TACACS+ server a start record accounting notice at the beginning of a process and a stop record at the end of the process.

```
ContentEngine(config)# aaa accounting exec default start-stop tacacs
```

Step 4  Activate accounting for all CLI commands at the normal privilege level (privilege level of 0) and indicate when accounting is to take place by using `aaa accounting commands 0 default {start-stop | stop-only | wait-start} tacacs` global configuration command.

In this example, the Content Engine is configured to record all CLI commands executed by an administrator who logged in to the Content Engine with an account that has normal privileges (privilege level of 0). The command configures the Content Engine to send the TACACS+ server a start record accounting notice at the beginning of a process and a stop record at the end of the process (the process being each CLI command that is executed by an administrator who has restricted privileges (privilege level of 0).

```
ContentEngine(config)# aaa accounting commands 0 default start-stop tacacs
```

Step 5  Activate accounting for all commands at the superuser privilege level and indicate when accounting is to take place by using the `aaa accounting commands 15 default {start-stop | stop-only | wait-start} tacacs` global configuration command.

In this example, the Content Engine is configured to record all CLI commands that are executed by a superuser. The command configures the Content Engine to send the TACACS+ server a start record accounting notice at the beginning of a process and a stop record at the end of the process (the process being each CLI command that is executed by a superuser (one with a privilege level of 15).

```
ContentEngine(config)# aaa accounting commands 15 default start-stop tacacs
```

Step 6  Verify the AAA accounting configuration.

```
ContentEngine# show aaa accounting
Accounting Type   Record event(s)  Protocol
------------------------------------------------------------------
Exec shell         start-stop        TACACS+
Command level 0     start-stop        TACACS+
Command level 15    start-stop        TACACS+
System             stop-only          TACACS+
```
Displaying the AAA Accounting Configuration for Standalone Content Engines

To display the current AAA configuration for a standalone Content Engine, enter the `show aaa accounting` EXEC command:

```
ContentEngine# show aaa accounting
Accounting Type  Record event(s)  Protocol
-------------------------------
Exec shell         unknown          unknown
Command level 0    unknown          unknown
Command level 15   unknown          unknown
System             start-stop       TACACS+
```

This command displays the AAA accounting configuration for the following accounting types:

- EXEC shell (accounting for EXEC processes [user shells])
- Command level for administrators with normal privileges (privilege level of 0)
- Command level for administrators with superuser privileges (privilege level of 15)
- System (accounting for all system-level events not associated with administrators, such as reloads)

Displaying and Clearing AAA Accounting Statistics on Standalone Content Engines

To display AAA accounting statistics on a standalone Content Engine, enter the `show statistics tacacs` EXEC command:

```
ContentEngine# show statistics tacacs
TACACS+ Statistics
-----------------------------------------------
Authentication:
Number of access requests:                 0
Number of access deny responses:           0
Number of access allow responses:          0

Authorization:
Number of authorization requests:          0
Number of authorization failure responses: 0
Number of authorization success responses: 0

Accounting:
Number of accounting requests:             0
Number of accounting failure responses:    0
Number of accounting success responses:    15
```

To clear the TACACS+ accounting statistics on the Content Engine, enter the `clear statistics tacacs` EXEC command.
Creating and Managing IP Access Control Lists for Standalone Content Engines

This chapter describes how to create and manage Internet Protocol (IP) access control lists (ACLs) on standalone Content Engines. This chapter contains the following sections:

- Introducing IP ACLs for Standalone Content Engines, page 19-2
- Understanding the Basics About Working with IP ACLs, page 19-4
- Defining and Activating IP ACLs on Standalone Content Engines, page 19-6
- Creating or Modifying IP ACLs on Standalone Content Engines, page 19-10
- Activating an IP ACL on an Interface, page 19-16
- Applying an IP ACL to an Application, page 19-17
- Deleting an IP ACL, page 19-23
- Viewing the Configuration of an IP ACL, page 19-23
- Clearing an IP ACL Counter, page 19-24

Note: Throughout this chapter, the term *IP ACLs* is used to refer to IP access control lists.
Introducing IP ACLs for Standalone Content Engines

In the ACNS 5.1 software and later releases, IP ACLs are supported. IP ACLs provide IP packet filtering. These IP ACLs provide a means to filter packets by allowing you to permit or deny IP packets from crossing specific interfaces on the Content Engine.

In environments that have standalone Content Engines, you may want to use this feature to control access to content services and management services on the Content Engine. For example, you can use IP ACLs to define a public interface on the Content Engine for content serving and a private interface for management services (for example, Telnet, Secure Shell (SSH), SNMP, HTTPS, and software upgrades). (See Figure 19-1.)

Figure 19-1 Using IP ACLs to Control Access to Specific Interfaces on a Standalone Content Engine

Note

In the ACNS 5.4.1 software and later releases, IP ACLs are also supported for nontransparent (proxy-mode requests) incoming FTP native requests and transparently-redirected incoming FTP native requests. For more information, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.
The following are some examples of how IP ACLs can be used in environments that have standalone Content Engines:

- A Content Engine resides on the customer premises and is managed by a service provider, and the service provider wants to secure the device for its management only.

- A Content Engine is deployed anywhere within the enterprise. As with routers and switches, the administrator wants to limit Telnet, SSH, and Content Engine GUI access to the IT source subnets.

- An application layer proxy firewall with a hardened outside interface has no ports exposed. (Hardened means that the interface carefully restricts which ports are available for access, primarily for security reasons. With an outside interface, many types of security attacks are possible.) The Content Engine’s outside address is Internet global, and its inside address is private. The inside interface has an IP ACL to limit Telnet, SSH, and Content Engine GUI access to the Content Engine.

- A Content Engine is deployed as a reverse proxy in an untrusted environment. The Content Engine administrator wants to allow only port 80 inbound traffic on the outside interface and outbound connections on the backend interface.

- A Content Engine using WCCP is positioned between a firewall and an Internet router or a subnet off the Internet router. Both the Content Engine and the router must have IP ACLs.

### Implementing IP ACLs for Standalone Content Engines

To implement IP ACLs, follow these steps:

**Step 1** Define the IP ACLs on the standalone Content Engine by using the `ip access-list` command.

**Step 2** Apply the defined IP ACL either inbound or outbound to an interface on the standalone Content Engine by using the `ip access-group` command.

**Note** You can also use IP ACLs to permit or deny Telnet, SSH, and SNMP access to this standalone Content Engine.

### Example of Defining and Activating IP ACLs

The following example shows how to define and activate an IP ACL on a standalone Content Engine. As the example shows, the first step is to use the `ip access-list` global configuration command to create an IP ACL for a standalone Content Engine. In this case, the IP ACL is named example and permits all web traffic but limits SSH access to a specific host:

```
ContentEngine(config)# ip access-list extended example
ContentEngine(config-ext-nacl)# permit tcp any any eq www
ContentEngine(config-ext-nacl)# permit tcp host 64.101.215.21 any eq ssh
ContentEngine(config-ext-nacl)# exit
```

After you create the IP ACL, use the `interface` global configuration command and the `ip access-group` configuration interface command to apply and activate the IP ACL for a specific interface on the Content Engine:

```
ContentEngine(config)# interface gigabitethernet 1/0
ContentEngine(config-if)# ip access-group example in
ContentEngine(config-if)# exit
```
After defining and activating the IP ACLs, view the running configuration on the Content Engine:

ContentEngine# show running-config
   
   interface GigabitEthernet 1/0
    ip address 10.1.1.50 255.255.0.0
    ip access-group example in
    exit
   .
   .
   
ip access-list extended example
    permit tcp any any eq www
    permit tcp host 10.101.215.21 any eq ssh
    exit
   .
   .

Note IP ACLs are defined for individual ACNS software devices only. IP ACLs cannot be managed globally across the ACNS network or through device groups. For information about using the Content Distribution Manager to create and manage IP ACLs on ACNS network devices (for example, a Content Engine that is registered with a Content Distribution Manager), see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.

For more background information about IP ACLs, see the next section, “Understanding the Basics About Working with IP ACLs.” For information about how to configure IP ACLs, see the “Defining and Activating IP ACLs on Standalone Content Engines” section on page 19-6.

**Understanding the Basics About Working with IP ACLs**

An IP ACL consists of one or more condition entries that specify the kind of packets that the Content Engine will drop or accept for further processing. The Content Engine applies each condition in the order in which it occurs in the IP ACL, which by default, is the order in which you configured the condition.

In the ACNS 5.1 software and later releases, there are two different types of IP ACLs:

- Standard ACLs
- Extended ACLs

Note The ACNS software CLI must be used to create and manage IP ACLs on a standalone Content Engine. The Content Engine GUI does not currently support the configuration of IP ACLs on a standalone Content Engine.
Working with Standard IP ACLs

Typically, standard ACLs are used for the following reasons:

- To allow connections from a host with a specific IP address
- To allow connections from hosts on a specific network

Accessing Standard IP ACL Configuration Mode

To work with standard IP ACLs, you must enter standard IP ACL configuration mode on a Content Engine. To access standard IP ACL configuration mode, enter the `ip access-list standard` global configuration command:

```
ContentEngine(config)# ip access-list standard {acl-name | acl-num}
```

- `acl-name` is the name of the standard IP ACL that you want to create or modify.
- `acl-num` is the number of the standard IP ACL that you want to create or modify.

After you enter standard IP ACL mode, the ContentEngine(config)# prompt changes to `ContentEngine(config-std-nacl)#`, where `nac1` stands for the specified standard access list.

For example, the following example shows how to enter standard IP ACL configuration mode in order to modify the standard IP ACL that has the ACL number of 2. The CLI enters the standard IP ACL configuration mode, in which all subsequent commands apply to the currently specified standard IP ACL (for example, the standard IP ACL nac2).

```
ContentEngine(config)# ip access-list standard 2
ContentEngine(config-std-nacl)#
```

Working with Extended IP ACLs

Extended IP ACLs generally use the following elements to control connections:

- Destination IP address
- IP protocol type
- UDP or TCP source or destination port
- ICMP message type or code
- TCP flag bits (established)

To create more restrictive conditions, these conditions can be combined with information about the source IP address. Table 19-3 lists the keywords that you can use to match specific Internet Control Message Protocol (ICMP) message types and codes.

Accessing Extended IP ACL Configuration Mode

To work with extended IP ACLs, you must enter extended IP ACL configuration mode on the Content Engine. To access extended IP ACL configuration mode, enter the `ip access-list extended` global configuration command:

```
ContentEngine(config)# ip access-list extended {acl-name | acl-num}
```

- `acl-name` is the name of the extended IP ACL that you want to create or modify.
- `acl-num` is the number of the extended IP ACL that you want to create or modify.
The following example shows how to enter extended IP ACL configuration mode in order to modify the extended IP ACL that has the ACL number of 101. The CLI enters extended IP ACL configuration mode where all subsequent commands apply to the currently specified extended IP ACL (for example, the extended IP ACL 101).

ContentEngine(config)\# ip access-list extended 101
ContentEngine(config-ext-nacl)#

Note: For information about how to create or modify an extended IP ACL, see the “Creating or Modifying IP ACLs on Standalone Content Engines” section on page 19-10.

### Defining and Activating IP ACLs on Standalone Content Engines

In some service provider deployments, a Content Engine can have one interface in the customer’s IP address space that serves content, and another interface in a private IP address space that the administrator uses for management purposes. The ACNS 5.1 software and later releases provide controls that allow various services to be associated with a particular interface (such as management services to the private IP address space) so that the enterprise customer can only access the Content Engine for serving content and not access it for management purposes.

To use IP ACLs in environments that have standalone Content Engines that are running the ACNS 5.1 software and later releases, the system administrator must complete the following tasks through the CLI:

1. **Define the IP ACLs by using the `ip access-list` command.**
2. **Apply an IP ACL to a specific interface on the Content Engine, by using the `interface` and `ip access-group` commands.**

Tip: Use the `ip access-group` command to apply an IP ACL to either inbound or outbound IP traffic on an interface.

### Usage Guidelines

When creating or modifying IP ACLs on a standalone Content Engine, remember the following important points:

- To create an entry in a standard or extended IP ACL, use the `deny` or `permit` keyword and specify the type of packets that you want the Content Engine to drop or to accept for further processing. By default, an access list denies everything because the list is terminated by an implicit `deny any` entry. Therefore, you must include at least one `permit` entry to create a valid access list.

Note: To allow connections from a specific network, use the `permit source-ip wildcard` command. Replace `source-ip` with a network ID or the IP address of any host on the network that you want to specify. Replace `wildcard` with the dotted decimal notation for a mask that is the reverse of a subnet mask, where a 0 indicates a position that must be matched and a 1 indicates a position that does not matter. For instance, the wildcard 0.0.0.255 causes the last 8 bits in the source IP address to be ignored. Therefore, the entry `permit 192.168.1.0 0.0.0.255` allows access from any host on the 192.168.1.0 network.
You can also apply an extended IP ACL to a specific application using the appropriate command. A reference to an IP ACL that does not exist is the equivalent of a **permit any** condition statement.

In the ACNS 5.4.1 software and later releases, you can use IP ACLs to grant or deny access to the native FTP proxy service that is running on a standalone Content Engine. To support this feature, the following two CLI commands were added in the ACNS 5.4.1 software release:

```
ftp-native access-list in {std-acl-num | std-acl-name}
ftp-native access-list out {ext-acl-num | ext-acl-name}
```

For more information, see the “Using IP ACLs to Control Native FTP Access” section on page 19-19.

In the ACNS 5.1 software and later releases, the SNMP and TFTP applications have a specific CLI command to configure the use of an IP ACL. The commands are as follows:

```
snmp-server access-list {std-acl-num | std-acl-name}
tftp-server access-list {std-acl-num | std-acl-name}
```

**Note** The **snmp-server access-list** and **tftp-server access-list** global configuration commands can only accept the name or number for a standard IP ACL and not an extended IP ACL.

Other application traffic (for example, Telnet and SSH) can be controlled by applying an IP ACL to an interface (typically, to inbound traffic) on the standalone Content Engine.

In the ACNS 5.2.1 software and later releases, use the **wccp access-list** global configuration command to specify an IP ACL that the Content Engine applies to WCCP GRE inbound traffic.

```
wccp access-list {acl-num | acl-name}
```

The WCCP access control list feature supports both standard and extended access control lists, and is not restricted to only standard access control lists, as is the case with SNMP and TFTP server access lists. For more information about configuring WCCP access control lists, see the “Configuring WCCP Access Lists for Standalone Content Engines” section on page 19-20.

For standard IP ACLs, the **wildcard** parameter of the **ip access-list** command is always optional. If the **host** keyword is specified for a standard IP ACL, then the **wildcard** parameter is not allowed, as shown in the following example:

```
ContentEngine(config)# ip access-list standard 1
ContentEngine(config-standard-nacl)# permit ?
    A.B.C.D Source address
    any Any source host
    host A single host address
ContentEngine(config-standard-nacl)# permit 10.1.1.1 ?
    A.B.C.D Source wildcard bits <== *** Wildcard parameter is optional here ***
    <cr>
ContentEngine(config-standard-nacl)# permit host 10.1.1.1 ? <== *** Wildcard parameter is not allowed here because the host keyword is used***
    <cr>
ContentEngine(config-standard-nacl)# permit 10.1.1.1
ContentEngine(config-standard-nacl)# exit
```
• For extended IP ACLs, the wildcard parameter is always required unless the host keyword is specified. If the host keyword is specified for an extended IP ACL, then the wildcard parameter is not allowed, as shown in the following example:

ContentEngine(config)# ip access-list extended 100
ContentEngine(config-ext-nacl)# permit ?
1-255> An IP Protocol Number
gre Cisco's GRE Tunneling
icmp Internet Control Message Protocol
ip Any IP Protocol
tcp Transport Control Protocol
udp User Datagram Protocol
ContentEngine(config-ext-nacl)# permit ip ?
A.B.C.D Source address
any Any source host
host A single host address
ContentEngine(config-ext-nacl)# permit ip 10.1.1.1 ?
A.B.C.D Source wildcard bits
<<<< *** Wildcard parameter is required here because the host keyword is not specified***
ContentEngine(config-ext-nacl)# permit ip host ?
A.B.C.D Source address
ContentEngine(config-ext-nacl)# permit ip host 10.1.1.1 ? <<== *** Wildcard parameter is not allowed here because the host keyword is used***
A.B.C.D Destination address
any Any destination host
host A single host address

• When you are in standard or extended IP ACL configuration mode, you can use the editing commands (list, delete, and move) to display entries, to delete a specific entry (a condition), or to change the order in which the entries will be evaluated.

ContentEngine(config)# ip access-list standard 1
ContentEngine(config-std-nacl)#?
delete Delete a condition
deny Specify packets to reject
ext Exit from this submode
insert Insert a condition
list List conditions
move Move a condition
no Negate a command or set its defaults
permit Specify packets to accept
ContentEngine(config-std-nacl)#

• To identify the line numbers that conditions map to, use the list command. This command lists the specified entries (or all entries when none are specified). Without this command, you would have to return to EXEC mode and then enter the show ip access-list EXEC command to obtain this mapping.

The following example shows how to use the list command.

ContentEngine(config-ext-nacl)# list
1 permit tcp host 10.1.1.1 any
2 permit tcp host 10.1.1.2 any
3 permit tcp host 10.1.1.3 any
ContentEngine(config-ext-nacl)#

• For information about how to delete an entire IP ACL from the Content Engine’s database, see the “Deleting an IP ACL” section on page 19-23.
Usage Guidelines for IP ACL Configuration Modes

When working with IP ACLs, remember the following important points about the IP ACL configuration modes:

- You must enter the standard IP ACL configuration mode to work with standard IP ACLs.

  ```
  ContentEngine(config)# ip access-list standard ?
  <1-99> Standard IP access-list number
  WORD Access-list name (max 30 characters)
  ```

- You must enter the extended IP ACL configuration mode to work with extended IP ACLs.

  ```
  ContentEngine(config)# ip access-list extended ?
  <100-199> Standard IP access-list number
  WORD Access-list name (max 30 characters)
  ```

Usage Guidelines for IP ACL Names

When creating IP ACL names, use the following naming guidelines:

- IP ACL names must be unique within the Content Engine

- When an IP ACL name is numeric (for example, **ip access-list standard acl-num** or **ip access-list extended acl-num**):
  
  - It can contain only numeric characters (for example, 101).
  
  - Numbers 1–99 represent standard IP ACLs.
  
  - Numbers 100–199 represent extended IP ACLs.

- When an IP ACL name is a word (for example, **ip access-list standard acl-name** or **ip access-list extended acl-name**):
  
  - It must begin with a nonnumeric character for example, snmpaccesslist).
  
  - It is limited to 30 characters.
  
  - It can contain the digits 0–9 within the string of characters (for example, snmpaccesslist7).
  
  - It can contain most of the printable special characters but no white space. The list of acceptable special characters includes the following: ~!@#$%^&*()_+-={}\[\]":;'<>.,/. The list of unacceptable special characters includes the following: `|"?.

**Note**

For more information about how to create or modify an IP ACL on a standalone Content Engine, see the next section, “Creating or Modifying IP ACLs on Standalone Content Engines.”
Creating or Modifying IP ACLs on Standalone Content Engines

To configure IP ACLs on a standalone Content Engine, follow these steps:

**Step 1** Access the Content Engine CLI in global configuration mode.

```
ContentEngine(config)#
```

**Step 2** From global configuration mode, access the appropriate IP ACL configuration mode, and specify the name or number of the IP ACL that you want to create, modify, or view.

- To create or modify a standard IP ACL, use the `ip access-list standard` global configuration command to enter into standard IP ACL configuration mode.

  ```
ip access-list standard {acl-name | acl-num}
  
  The following example shows how to create or modify a standard IP ACL that has the ACL number of 59:
  
  ContentEngine(config)# ip access-list standard 59
  ```

  The CLI enters standard IP ACL configuration mode, in which all subsequent commands apply to the current standard IP ACL, and the following prompt appears:

  ```
  ContentEngine(config-std-nacl)#
  ```

- To create or modify an extended IP ACL, use the `ip access-list extended` command to enter into extended IP ACL configuration mode.

  ```
ip access-list extended {acl-name | acl-num}
  
  The following example shows how to create or modify an extended IP ACL named test2 by specifying its name:
  
  ContentEngine(config)# ip access-list extended test2
  ```

  The CLI enters extended IP ACL configuration mode, in which all subsequent commands apply to the current extended IP ACL, and the following prompt appears:

  ```
  ContentEngine(config-ext-nacl)#
  ```

**Step 3** To add, delete, or modify conditions in a standard ACL, enter the following commands from the standard IP ACL configuration mode:

- **a.** To add a line to the standard IP ACL, use the following syntax.

  For example, choose a purpose (permit or deny) that specifies whether a packet is to be passed or dropped, enter the source IP address, and enter the source IP wildcard address.

  ```
  [insert line-num] {deny | permit} {source-ip [wildcard] | host source-ip | any}
  ```

- **b.** To delete a line from the standard IP ACL, use the following syntax:

  ```
  delete line-num
  ```

- **c.** To move a line to a new position within the standard IP ACL, use the following syntax:

  ```
  move old-line-num new-line-num
  ```

**Note** For a list of extended IP ACL conditions, see Table 19-4.
Step 4  To add, delete, or modify conditions in an extended ACL, enter the following commands from extended IP ACL configuration mode:

a. To delete a line from the extended IP ACL, use the following syntax:

   delete line-num

b. To move a line to a new position within the extended IP ACL, use the following syntax:

   move old-line-num new-line-num

c. To add a condition to the extended IP ACL, enter the options according to the protocol you choose:

- For IP, use the following syntax to add a condition:

  \[
  \text{insert line-num} \{\text{deny | permit}\} \{\text{gre | ip | proto-num}\} \{\text{source-ip wildcard | host source-ip | any}\} \{\text{dest-ip wildcard | host dest-ip | any}\}
  \]

- For TCP, use the following syntax to add a condition:

  \[
  \text{insert line-num} \{\text{deny | permit}\} \text{tcp} \{\text{source-ip wildcard | host source-ip | any}\} \{\text{dest-ip wildcard | host dest-ip | any}\} \{\text{operator port [port]} \{\text{established}\}
  \]

- For UDP, use the following syntax to add a condition:

  \[
  \text{insert line-num} \{\text{deny | permit}\} \text{udp} \{\text{source-ip wildcard | host source-ip | any}\} \{\text{dest-ip wildcard | host dest-ip | any}\} \{\text{operator port [port]} \}
  \]

- For ICMP, use the following syntax to add a condition:

  \[
  \text{insert line-num} \{\text{deny | permit}\} \text{icmp} \{\text{source-ip wildcard | host source-ip | any}\} \{\text{dest-ip wildcard | host dest-ip | any}\} \{\text{icmp-type [code] | icmp-msg}\}
  \]

Note  For extended IP ACLs, the wildcard parameter is required if the host keyword is not specified. For a list of the keywords that you can use to match specific ICMP message types and codes, see Table 19-3. For a list of supported UDP and TCP keywords, see Table 19-1 and Table 19-2. For a list of extended IP ACL conditions, see Table 19-5.

Step 5  To add another condition to a standard IP ACL, repeat Step 3. To add another condition (entry) to an extended IP ACL, repeat Step 4.

Step 6  Activate and apply this IP ACL to a specific interface on this Content Engine by using the interface and ip access-group commands.
For more information about activating and applying an IP ACL to a specific interface, see the “Activating an IP ACL on an Interface” section on page 19-16 and the “Applying an IP ACL to an Application” section on page 19-17.

---

### List of Keywords for Extended IP ACLs

Table 19-1 lists the UDP keywords that you can use with extended IP ACLs.

**Table 19-1**

<table>
<thead>
<tr>
<th>CLI Keyword</th>
<th>Description</th>
<th>UDP Port Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>bootpc</td>
<td>Bootstrap Protocol (BOOTP) client</td>
<td>68</td>
</tr>
<tr>
<td>bootps</td>
<td>Bootstrap Protocol (BOOTP) server</td>
<td>67</td>
</tr>
<tr>
<td>domain</td>
<td>Domain Name Service (DNS)</td>
<td>53</td>
</tr>
<tr>
<td>mms</td>
<td>Microsoft Media Server Protocol</td>
<td>1755</td>
</tr>
<tr>
<td>netbios-dgm</td>
<td>NetBIOS datagram service</td>
<td>138</td>
</tr>
<tr>
<td>netbios-ns</td>
<td>NetBIOS name service</td>
<td>137</td>
</tr>
<tr>
<td>netbios-ss</td>
<td>NetBIOS session service</td>
<td>139</td>
</tr>
<tr>
<td>nfs</td>
<td>Network File Server service</td>
<td>2049</td>
</tr>
<tr>
<td>ntp</td>
<td>Network Time Protocol</td>
<td>123</td>
</tr>
<tr>
<td>snmp</td>
<td>Simple Network Management Protocol</td>
<td>161</td>
</tr>
<tr>
<td>snmptrap</td>
<td>SNMP traps</td>
<td>162</td>
</tr>
<tr>
<td>tacacs</td>
<td>Terminal Access Controller (TAC) Access Control System</td>
<td>49</td>
</tr>
<tr>
<td>tftp</td>
<td>Trivial File Transfer Protocol</td>
<td>69</td>
</tr>
<tr>
<td>wccp</td>
<td>Web Cache Communication Protocol</td>
<td>2048</td>
</tr>
</tbody>
</table>

Table 19-2 lists the TCP keywords that you can use with extended IP ACLs.

**Table 19-2**

<table>
<thead>
<tr>
<th>CLI Keyword</th>
<th>Description</th>
<th>TCP Port Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Domain Name Service</td>
<td>53</td>
</tr>
<tr>
<td>exec</td>
<td>Exec (RCP)</td>
<td>512</td>
</tr>
<tr>
<td>ftp</td>
<td>File Transfer Protocol</td>
<td>21</td>
</tr>
<tr>
<td>ftp-data</td>
<td>FTP data connections (used infrequently)</td>
<td>20</td>
</tr>
<tr>
<td>https</td>
<td>Secure HTTP</td>
<td>443</td>
</tr>
<tr>
<td>nfs</td>
<td>Network File Server service</td>
<td>2049</td>
</tr>
<tr>
<td>rtsp</td>
<td>Real-Time Streaming Protocol</td>
<td>554</td>
</tr>
</tbody>
</table>
Table 19-2  TCP Keywords and Port Numbers (continued)

<table>
<thead>
<tr>
<th>CLI Keyword</th>
<th>Description</th>
<th>TCP Port Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh</td>
<td>Secure Shell login</td>
<td>22</td>
</tr>
<tr>
<td>tacacs</td>
<td>Terminal Access Controller (TAC) Access Control System</td>
<td>49</td>
</tr>
<tr>
<td>telnet</td>
<td>Telnet</td>
<td>23</td>
</tr>
<tr>
<td>www</td>
<td>World Wide Web (HTTP)</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 19-3 lists the keywords that you can use to match specific ICMP message types and codes.

Table 19-3  Keywords for ICMP Message Type and Code

<table>
<thead>
<tr>
<th>CLI Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>administratively-prohibited</td>
<td>alternate-address</td>
</tr>
<tr>
<td>conversion-error</td>
<td>dod-host-prohibited</td>
</tr>
<tr>
<td>dod-host-prohibited</td>
<td>echo</td>
</tr>
<tr>
<td>dod-net-prohibited</td>
<td>general-parameter-problem</td>
</tr>
<tr>
<td>echo-reply</td>
<td>host-precedence-unreachable</td>
</tr>
<tr>
<td>host-isolated</td>
<td>host-tos-redirect</td>
</tr>
<tr>
<td>host-redirect</td>
<td>host-unknown</td>
</tr>
<tr>
<td>host-tos-redirect</td>
<td>information-reply</td>
</tr>
<tr>
<td>host-unreachable</td>
<td>mask-reply</td>
</tr>
<tr>
<td>information-request</td>
<td>mobile-redirect</td>
</tr>
<tr>
<td>mask-request</td>
<td>net-tos-redirect</td>
</tr>
<tr>
<td>net-redirect</td>
<td>net-tos-unreachable</td>
</tr>
<tr>
<td>net-tos-unreachable</td>
<td>net-unknown</td>
</tr>
<tr>
<td>network-unknown</td>
<td>no-room-for-option</td>
</tr>
<tr>
<td>option-missing</td>
<td>packet-too-big</td>
</tr>
<tr>
<td>parameter-problem</td>
<td>port-unreachable</td>
</tr>
<tr>
<td>precedence-unreachable</td>
<td>protocol-unreachable</td>
</tr>
<tr>
<td>reassembly-timeout</td>
<td>redirect</td>
</tr>
<tr>
<td>router-advertisement</td>
<td>router-solicitation</td>
</tr>
<tr>
<td>source-quench</td>
<td>source-route-failed</td>
</tr>
<tr>
<td>time-exceeded</td>
<td>timestamp-reply</td>
</tr>
<tr>
<td>timestamp-request</td>
<td>traceroute</td>
</tr>
<tr>
<td>ttl-exceeded</td>
<td>unreachable</td>
</tr>
</tbody>
</table>
IP ACL Conditions

Table 19-4 describes the standard IP ACL conditions.

Table 19-4  Standard IP ACL Conditions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>(Optional) Inserts the conditions following the specified line number into the standard IP ACL.</td>
</tr>
<tr>
<td>line-num</td>
<td>Identifies the entry at a specific line number in the standard IP ACL.</td>
</tr>
<tr>
<td>deny</td>
<td>Causes packets that match the specified conditions to be dropped.</td>
</tr>
<tr>
<td>permit</td>
<td>Causes packets that match the specified conditions to be accepted for further processing.</td>
</tr>
<tr>
<td>source-ip</td>
<td>Source IP address. The number of the network or host from which the packet is being sent, specified as a 32-bit quantity in 4-part dotted decimal format (for example, 0.0.0.0).</td>
</tr>
<tr>
<td>wildcard</td>
<td>Specifies the portions of the preceding IP address to match, expressed using 4-digit, dotted-decimal notation. Bits to match are identified by a digital value of 0; bits to ignore are identified by a 1. For standard IP ACLs, the wildcard parameter of the ip access-list command is always optional. If the host keyword is specified for a standard IP ACL, then the wildcard parameter is not allowed.</td>
</tr>
<tr>
<td>host</td>
<td>Matches the following IP address.</td>
</tr>
<tr>
<td>any</td>
<td>Matches any IP address.</td>
</tr>
<tr>
<td>delete</td>
<td>Deletes the specified entry (condition) from the standard IP ACL.</td>
</tr>
<tr>
<td>line-num</td>
<td>Identifies the entry at a specific line number in the standard IP ACL.</td>
</tr>
<tr>
<td>list</td>
<td>Lists the specified entries (or all entries when none are specified).</td>
</tr>
<tr>
<td>start-line-num</td>
<td>(Optional) Starting line number of the list.</td>
</tr>
<tr>
<td>end-line-num</td>
<td>(Optional) Ending line number of the list.</td>
</tr>
<tr>
<td>move</td>
<td>Moves the specified entry in the standard IP ACL to a new position in the list.</td>
</tr>
<tr>
<td>old-line-num</td>
<td>Specifies the line number of the entry to move.</td>
</tr>
<tr>
<td>new-line-num</td>
<td>Specifies the new position of the entry. The existing entry is moved to this new position in the standard IP ACL.</td>
</tr>
</tbody>
</table>

Table 19-5 describes the extended IP ACL conditions.

Table 19-5  Extended IP ACL Conditions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>(Optional) Inserts the conditions at the specified line number into the extended IP ACL.</td>
</tr>
<tr>
<td>line-num</td>
<td>Identifies the entry at a specific line number in the extended IP ACL.</td>
</tr>
<tr>
<td>deny</td>
<td>Causes packets that match the specified conditions to be dropped.</td>
</tr>
<tr>
<td>permit</td>
<td>Causes packets that match the specified conditions to be accepted for further processing.</td>
</tr>
<tr>
<td>source-ip</td>
<td>Source IP address.</td>
</tr>
<tr>
<td>wildcard</td>
<td>Specifies the portions of the preceding IP address to match, expressed using 4-digit, dotted-decimal notation. Bits to match are identified by a digital value of 0; bits to ignore are identified by a 1. For extended IP ACLs, the wildcard parameter is always required unless the host keyword is specified. If the host keyword is specified for an extended IP ACL, then the wildcard parameter is not allowed.</td>
</tr>
</tbody>
</table>
Creating or Modifying IP ACLs on Standalone Content Engines

Table 19-5  Extended IP ACL Conditions (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Matches the following IP address.</td>
</tr>
<tr>
<td>any</td>
<td>Matches any IP address.</td>
</tr>
<tr>
<td>gre</td>
<td>Matches packets using the generic routing encapsulation (GRE) protocol.</td>
</tr>
<tr>
<td>ip</td>
<td>Matches all IP packets.</td>
</tr>
<tr>
<td>proto-num</td>
<td>Specifies the IP protocol number.</td>
</tr>
<tr>
<td>tcp</td>
<td>Matches packets using the TCP protocol.</td>
</tr>
<tr>
<td>udp</td>
<td>Matches packets using the UDP protocol.</td>
</tr>
<tr>
<td>operator</td>
<td>(Optional) Specifies the operator to use with specified ports, where ( \text{lt} ) = less than, ( \text{gt} ) = greater than, ( \text{eq} ) = equal to, ( \text{neq} ) = not equal to, and ( \text{range} ) = an inclusive range. The following shows an example of an extended IP ACL that uses the \text{equal to} \ operator.</td>
</tr>
<tr>
<td>port</td>
<td>(Optional) Specifies the port, using a number (0–65535) or a keyword; 2 port numbers are required with the \text{range} \ operator. Use any of the following keywords with TCP: \text{domain}, \text{exec}, \text{ftp}, \text{ftp-data}, \text{https}, \text{mms}, \text{nfs}, \text{rtsp}, \text{ssh}, \text{tacacs}, \text{telnet}, and \text{www}. Use any of the following keywords with UDP: \text{bootpc}, \text{bootps}, \text{domain}, \text{mms}, \text{netbios-dgm}, \text{netbios-ns}, \text{netbios-ss}, \text{nfs}, \text{ntp}, \text{snmp}, \text{snmptrap}, \text{tacacs}, \text{tftp}, and \text{wccp}. For example:</td>
</tr>
<tr>
<td>dest-ip</td>
<td>Destination IP address.</td>
</tr>
<tr>
<td>established</td>
<td>(Optional) Matches TCP packets with the ACK or RST bits set.</td>
</tr>
<tr>
<td>icmp</td>
<td>Matches ICMP packets.</td>
</tr>
<tr>
<td>icmp-type</td>
<td>(Optional) Matches by ICMP message type, expressed as a number (0–255).</td>
</tr>
<tr>
<td>code</td>
<td>(Optional) Used with \text{icmp-type} to further match by ICMP code type, expressed as a number from 0 to 255.</td>
</tr>
<tr>
<td>icmp-msg</td>
<td>(Optional) Matches using a combination of ICMP message type and code types, as expressed by the keywords listed in Table 19-3.</td>
</tr>
<tr>
<td>delete</td>
<td>Deletes the specified entry (condition) from the extended IP ACL.</td>
</tr>
<tr>
<td>line-num</td>
<td>Identifies the entry at a specific line number in the extended IP ACL.</td>
</tr>
<tr>
<td>list</td>
<td>Lists the specified entries (or all entries when none are specified).</td>
</tr>
<tr>
<td>start-line-num</td>
<td>(Optional) Starting line number of the list.</td>
</tr>
<tr>
<td>end-line-num</td>
<td>(Optional) Ending line number of the list.</td>
</tr>
<tr>
<td>move</td>
<td>Moves the specified entry in the list to a new position in the list.</td>
</tr>
<tr>
<td>old-line-num</td>
<td>Specifies the line number of the entry to move.</td>
</tr>
<tr>
<td>new-line-num</td>
<td>Specifies the new position of the entry. The existing entry is moved to this position in the access list.</td>
</tr>
<tr>
<td>exit</td>
<td>Returns to the CLI global configuration mode prompt.</td>
</tr>
</tbody>
</table>
Activating an IP ACL on an Interface

In the ACNS 5.1 software and later releases, there are controls that allow various services to be tied to a particular interface. To activate an IP ACL on a particular interface on a standalone Content Engine, use the `ip access-group` interface configuration command. You can use one outbound IP ACL and one inbound IP ACL on each interface.

Before entering the `ip access-group` command, enter interface configuration mode for the interface to which you want to apply the IP ACL.

The following commands apply and activate the IP ACL named acl-out pm outbound traffic on the FastEthernet interface slot 0/port 0:

```
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# ip access-group acl-out out
```

The following commands apply and activate the IP ACL named example on the inbound traffic on the Gigabit Ethernet interface on port 1/slot 0:

```
ContentEngine(config)# interface gigabitethernet 1/0
ContentEngine(config-if)# ip access-group example in
ContentEngine(config-if)# exit
```

To apply and activate an IP ACL on a specific interface on the standalone Content Engine, follow these steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter interface configuration mode for the interface to which you want to apply the IP ACL. For example, the following example shows how to enter interface configuration mode for the Fast Ethernet interface on slot 0/port 0 of the Content Engine:</td>
</tr>
<tr>
<td></td>
<td>ContentEngine(config)# interface FastEthernet 0/0 ContentEngine(config-if)#</td>
</tr>
<tr>
<td>2</td>
<td>Apply the predefined IP ACL to the specified interface. For example, the following example shows how apply the predefined IP ACL named acl-out to outbound traffic on the FastEthernet interface slot 0/port 0:</td>
</tr>
<tr>
<td></td>
<td>ContentEngine(config-if)# ip access-group acl-out out</td>
</tr>
</tbody>
</table>

Table 19-6 describes the parameters for the `ip access-group` command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl-name</td>
<td>Alphanumeric identifier up to 30 characters, beginning with a letter that identifies the IP ACL to apply to the current interface.</td>
</tr>
<tr>
<td>acl-num</td>
<td>Numeric identifier that identifies the IP ACL to apply to the current interface (1–99 for standard IP ACLs; 100–199 for extended IP ACLs).</td>
</tr>
<tr>
<td>in</td>
<td>Applies the specified IP ACL to inbound packets on the current interface.</td>
</tr>
<tr>
<td>out</td>
<td>Applies the specified IP ACL to outbound packets on the current interface.</td>
</tr>
</tbody>
</table>
Applying an IP ACL to an Application

In the ACNS 5.4.1 software and later releases, FTP has the following specific CLI configuration command to configure the use of an IP ACL for FTP native proxy-mode and transparently redirected (WCCP-redirected) connections:

```
ftp-native access-list in {std-acl-num | std-acl-name}
ftp-native access-list out {ext-acl-num | ext-acl-name}
```

Use the `ftp-native access-list in` global configuration command to configure a standard ACL for the following types of inbound FTP native connections: proxy mode connections and transparently redirected (WCCP-redirected) connections. For standard ACLs, access control will be based on the source address of the FTP client that sent the native FTP request.

Use the `ftp-native access-list out` global configuration command to configure an extended ACL for the following types of outbound FTP native connections: proxy mode connections and transparently redirected (WCCP-redirected) connections. For extended ACLs, access control will be based on the source and destination addresses.

In the ACNS 5.1 software and later releases, SNMP and TFTP have the following specific CLI configuration commands to configure the use of an IP ACL:

```
snmp-server access-list {std-acl-num | std-acl-name}
tftp-server access-list {std-acl-num | std-acl-name}
```

**Note** The `snmp-server access-list` and `tftp-server access-list` global configuration commands can only accept the name or number for a standard IP ACL and not an extended IP ACL.

In the ACNS 5.2.1 software and later releases, specify an IP access list that the Content Engine applies to inbound WCCP GRE encapsulated traffic that it receives by using the `wccp access-list` global configuration command.

```
wccp access-list {acl-num | acl-name}
```

The WCCP access list feature supports both standard and extended access lists, and is not restricted to only standard access lists, as is the case with SNMP and TFTP server access lists.

Other application traffic (for example, Telnet and SSH) can be controlled by applying an IP ACL to an interface (typically to inbound traffic) on the standalone Content Engine.

**Note** In the ACNS 5.1 software and later releases, you need to use the `ip access-list` global configuration command to allow or deny access to the user through the TFTP protocol. Unless the IP ACL is configured for the TFTP service, the security of the content can be at risk, and TFTP will not function properly.

In the ACNS 5.0 software, TFTP access was denied to the user by default. To allow access to the user, the administrator had to use the `trusted-host` command. The `trusted-host` command has been deprecated in the ACNS 5.1 software. Consequently, if the `trusted-host` command is used on Content Engines that are running an ACNS software release earlier than Release 5.1, and the device is subsequently upgraded to the ACNS software 5.1 and later releases, the `trusted-host` command shows up in the CLI but does not have any effect on the TFTP protocol. The trusted host configurations can be deleted by using the `no trusted-host` command.
Using IP ACLs to Control SNMP Access

To use standard IP ACLs to control access to the SNMP agent on a standalone Content Engine, follow these steps:

**Step 1** Create an IP ACL to control access to the SNMP agent on the Content Engine by using the `ip access-list standard` command.

**Step 2** Associate this IP ACL with the SNMP server (the standalone Content Engine), and activate this standard IP ACL on the Content Engine.

```bash
ContentEngine(config)# snmp-server access-list {std-acl-num | std-acl-name}
```

- `std-acl-name` is the name of the standard IP ACL that you want to associate with this Content Engine.
- `std-acl-num` is the number of the standard IP ACL that you want to associate with this Content Engine.

The SNMP agent on the Content Engine checks against the specified IP ACL (for example, ACL 1) before accepting or dropping incoming packets.

```bash
ContentEngine(config)# snmp-server access-list 1
```

Using IP ACLs to Control TFTP Access

To use standard IP ACLs to control access to the TFTP service on a standalone Content Engine, follow these steps:

**Step 1** Create an access list to control access to the TFTP service that is running on the standalone Content Engine.

For example, the following commands define an access list that permits access to the TFTP service for TFTP clients on the 192.168.1.0 subnetwork:

```bash
ContentEngine(config)# ip access-list standard 2
ContentEngine(config-std-nacl)# permit 192.168.1.0 0.0.0.255
ContentEngine(config-std-nacl)# exit
ContentEngine(config)#
```

**Step 2** Associate this IP ACL (ACL 2) with the TFTP server (the standalone Content Engine), and activate this standard IP ACL on the Content Engine. The Content Engine checks against the specified access control list before permitting or denying access to the TFTP service that it is running.

The following example configures the Content Engine to check against access control list 2 before permitting or denying TFTP access to a user (a TFTP client).

```bash
ContentEngine(config)# tftp-server access-list 2
```
Using IP ACLs to Control Native FTP Access

In the ACNS 5.4.1 software and later releases, you can use IP ACLs to permit or deny access to the native FTP proxy service that is running on a standalone Content Engine. This support is available for the following types of incoming native FTP requests:

- Nontransparent FTP native requests
- Transparent FTP native requests (incoming FTP native requests that are transparently intercepted and redirected to a Content Engine by a WCCP-enabled router)

To support this new feature, the following two CLI commands were added in the ACNS 5.4.1 software release:

```
ftp-native access-list in {std-acl-num | std-acl-name}
ftp-native access-list out {ext-acl-num | ext-acl-name}
```

For a standard IP ACL, access control for the incoming FTP native request is based on the source address. For an extended IP ACL, access control for the FTP native requests that are outbound to the origin server is based on the source and destination addresses.

To use IP ACLs to control access to the native FTP proxy service that is running on a standalone Content Engine, follow these steps:

**Step 1** Create an access list to control access to the native FTP proxy service that is running on the standalone Content Engine.

For example, the following commands define an access list that permits access to the native FTP proxy service for FTP clients on the 192.168.1.0 subnetwork:

```
ContentEngine(config)# ip access-list standard 3
ContentEngine(config-std-nacl)# permit 192.168.1.0 0.0.0.255
ContentEngine(config-std-nacl)# exit
ContentEngine(config)#
```

**Step 2** Associate this IP ACL (ACL 3) with the native FTP proxy service, and activate this standard IP ACL on the Content Engine. The Content Engine checks against the specified access control list before permitting or denying access to the native FTP proxy service.

The following example configures the Content Engine to check against access control list 3 before permitting or denying FTP proxy access to a user (for example, such FTP clients as a Reflection X client, a WS-FTP client, or a UNIX or DOS command line FTP program):

```
ContentEngine(config)# ftp-native access-list in 3
```

**Step 3** (Optional). Configure customized response messages that the Content Engine will send to an FTP client in response to an incoming proxy-mode connection.

In the ACNS 5.4.1 software and later releases, you can use the `ftp-native custom-message` global configuration command to configure customized response messages, which the Content Engine sends to an FTP client in response to an incoming proxy mode connection. You can use the `ftp-native custom-message` EXEC command to create, upload, and download a file that contains one of the following custom messages:

- A custom welcome message for welcoming proxy-mode connections from FTP clients
- A custom error message for denying a user (an FTP client) access to the native FTP proxy service based on the configured IP ACLs for the native FTP proxy service

For more information about this topic, see the “Creating Custom Messages for FTP Proxy Responses for FTP Native Requests” section on page 5-19.
Chapter 19  Creating and Managing IP Access Control Lists for Standalone Content Engines

Applying an IP ACL to an Application

Step 4  From an FTP client, issue a native FTP request to the Content Engine to verify that the defined access control list works properly.

The following example issues a native FTP request from an FTP client that is on the 192.168.1.0 subnetwork. In this case, the FTP client is a UNIX command line FTP program and the Content Engine has an IP address of 10.1.1.50. Because the source address (the FTP client’s IP address) is in the 192.168.1.0 subnetwork, this FTP client is granted access to the FTP service and the Content Engine displays the welcome message to the FTP client:

```shell
shell# ftp -d 10.1.1.50 8501
Connected to 10.1.1.50
220 Welcome to FTP-proxy. Login to the proxy using username and password.
Name (10.1.1.50:admin):
```

Using IP ACLs to Control WCCP Access

To use standard or extended IP ACLs to control WCCP access on a standalone Content Engine, follow these steps:

Step 1  Create a standard or extended IP ACL to control WCCP access on the Content Engine.

- To create a standard IP ACL, use the `ip access-list standard` command.
- To create an extended IP ACL, use the `ip access-list extended` command.

Step 2  Associate the IP ACL with the Content Engine, and activate this IP ACL on the Content Engine.

```
ContentEngine(config)# wccp access-list {acl-name | acl-number}
```

- `acl-name` is the name of the standard or extended IP ACL that you want to associate with this Content Engine.
- `acl-number` is the number of the standard or extended IP ACL that you want to associate with this Content Engine.

The Content Engine applies the specified IP ACL (for example, ACL 2) to WCCP GRE inbound traffic.

```
ContentEngine(config)# wccp access-list 2
```

Configuring WCCP Access Lists for Standalone Content Engines

In the ACNS 5.2.1 software and later releases, specify an IP access list that the Content Engine applies to WCCP GRE encapsulated traffic that it receives inbound by using the `wccp access-list` global configuration command.

```
wccp access-list {acl-name | acl-number}
```

`acl-name` or `acl-number` represent either a standard or an extended IP access list.

The default is that no WCCP access list is configured; therefore, the WCCP access lists are not shown as part of the Content Engine’s configuration.
The following example shows sample output from the `show ip access-list` EXEC command if a WCCP access list has been configured on a Content Engine:

```
Content Engine# show ip access-list
Space available:
  48 access lists
  497 access list conditions
Standard IP access list test
  1 permit 10.1.1.1
    (implicit deny any:0 matches)
total invocations:0
Extended IP access list no_www.linux.org
  1 deny tcp any host 10.1.1.1 (29 matches)
  2 permit ip any any (30 matches)
    (implicit fragment permit:0 matches)
    (implicit deny ip any any:0 matches)
total invocations:59
```

Interface access list references:
- GigabitEthernet 2/0  inbound  pc_test (Not Defined)

Application access list references:
- snmp-server                     standard  test
- tftp_server                     standard  test4
- UDP ports: none
- tftp_server                     standard  test4
- UDP ports:   69 (List Not Defined)
- WCCP                            either    no_www.linux.org
- Any IP Protocol

Content Engine#

In the ACNS 5.2.1 software and later releases, the output of the `show wccp gre` EXEC command includes two counters that are related to the WCCP access list feature:

```
----------------
Packets w/WCCP GRE received too small:     0
Packets dropped due to IP access-list deny:29
----------------
```

The first counter represents the number of properly encapsulated WCCP GRE packets that were dropped because they were too short to contain a complete IP packet header.

The second counter represents the number of packets that were dropped because they were denied by the specified WCCP access list.

The following example shows sample output from the `show wccp gre` EXEC command when WCCP access lists are defined on the Content Engine:

```
Content Engine# show wccp gre
Transparent GRE packets received:          366
Transparent non-GRE packets received:      0
Transparent non-GRE packets passed through:0
Total packets accepted:                    337
Invalid packets received:                  0
Packets received with invalid service:     0
Packets received on a disabled service:    0
Packets received too small:                0
Packets dropped due to zero TTL:           0
Packets dropped due to bad buckets:        0
Packets dropped due to no redirect address:0
Connections bypassed due to load:          0
Packets sent back to router:               0
Packets sent to another CE:                0
GRE fragments redirected:                  0
```
Chapter 19      Creating and Managing IP Access Control Lists for Standalone Content Engines

Applying an IP ACL to an Application

You can also display the above output by entering the show statistics wccp gre EXEC command.

Configuration Examples

The following example shows how to use the ip access-list extended global configuration command to create an extended IP ACL named example. This extended IP ACL is created to allow all web traffic but to only allow a specific host (host 10.1.1.5) administrative access using SSH.

ContentEngine(config)# ip access-list extended example
ContentEngine(config-ext-nacl)# permit tcp any any eq www
ContentEngine(config-ext-nacl)# permit tcp host 10.1.1.5 any eq ssh
ContentEngine(config-ext-nacl)# exit

The following example shows a standalone Content Engine that has been configured to use interface access lists and application access lists:

ContentEngine# show ip access-list
Space available:
  47 access lists
  492 access list conditions

  Standard IP access list 1
  1 permit 10.1.1.2
  2 deny 10.1.2.1
  (implicit deny any: 2 matches)
  total invocations: 2

  Extended IP access list 100
  1 permit tcp host 10.1.1.1 any
  2 permit tcp host 10.1.1.2 any
  3 permit tcp host 10.1.1.3 any
  (implicit fragment permit: 0 matches)
  (implicit deny ip any any: 0 matches)
  total invocations: 0

  Standard IP access list test
  1 permit 1.1.1.1 (10 matches)
  2 permit 1.1.1.3
  3 permit 1.1.1.2
  (implicit deny ip any any: 2 matches)
  total invocations: 12

Interface access list references:
  FastEthernet 0/0 inbound 100

Application access list references:
  tftp_server  standard 1
  UDP ports: 69
Deleting an IP ACL

To delete an IP ACL (including all conditions and references in network interfaces and applications) from the Content Engine database, follow these steps:

---

**Step 1** Access the Content Engine CLI in global configuration mode.

```
ContentEngine(config)#
```

**Step 2** Specify the name or number of the IP ACL that you want to delete.

- To delete a standard IP ACL, specify the standard IP ACL that you want to delete.

  ```
  ContentEngine(config)# no ip access-list standard {acl-name | acl-num}
  ```

  The following example shows how to delete the standard IP ACL named test2:

  ```
  ContentEngine(config)# no ip access-list standard test2
  ```

- To delete an extended IP ACL, specify the extended IP ACL that you want to delete.

  ```
  ContentEngine(config)# no ip access-list extended {acl-name | acl-num}
  ```

  The following example shows how to delete the extended IP ACL named example:

  ```
  ContentEngine(config)# no ip access-list extended example
  ```

---

Viewing the Configuration of an IP ACL

To view the configuration of the IP ACLs currently defined on a Content Engine, use the `show ip access-list` EXEC command:

```
show ip access-list [acl-name | acl-num]
```

The `show ip access-list` EXEC command allows you to display configuration information about the IP ACLs that have been defined on the current system (in this case, the standalone Content Engine). Unless you identify a specific IP ACL by name or number, the system displays information about all the defined IP ACLs, including the following sections:

- Available space for new lists and conditions
- Defined IP ACLs
- References by interface and application

The following example shows sample output from the `show ip access-list` EXEC command when a specific IP ACL is not specified:

```
ContentEngine# show ip access-list
Space available:
    47 access lists
    492 access list conditions

Standard IP access list 1
1 permit 10.1.1.2
2 deny   10.1.2.1
     (implicit deny any: 2 matches)
total invocations: 2
Extended IP access list 100
```
Clearing an IP ACL Counter

To clear the IP ACL counter and to reset the statistical information for an IP ACL on a Content Engine, use the `clear ip access-list counter` EXEC command:

```bash
ContentEngine# clear ip access-list counters {acl-name | acl-num}
```

Use this EXEC command to clear the IP ACL counters associated with the condition statements of all the existing IP ACLs. If you specify an IP ACL name or number, then only the specified list’s counters will be cleared.

---

**Note**

The system displays the number of packets that have matched a condition statement only if the number is greater than zero.
Viewing and Modifying TCP Stack Parameters on Standalone Content Engines

This chapter describes TCP stack parameters and how best to optimize them for caching purposes on standalone Content Engines. It explains how to view or modify TCP stack parameters, and contains the following sections:

- TCP Stack Overview, page 20-2
- Viewing or Modifying TCP Parameters on Standalone Content Engines, page 20-2
- Displaying TCP Configuration Information, page 20-5
- TCP-Over-Satellite Extensions, page 20-6

Note

For complete syntax and usage information for the CLI commands used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication.
TCP Stack 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 that determine what content is viewed (content filtering)
- To increase the throughput of HTTP streams over TCP end to end

Another reason is to fine tune the TCP stack parameters to improve 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.

The relevant TCP parameters to maximize cache performance and throughput include the ability to tune timeout periods, client and server receive and 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.

Viewing or Modifying TCP Parameters on Standalone Content Engines

You can use the Content Engine GUI or CLI to view or modify TCP parameters on a standalone Content Engine.

- From the Content Engine GUI, choose **System > TCP**. Use the displayed TCP window to view or modify TCP parameters for this Content Engine. The existing TCP parameters are displayed in the TCP window. To modify a TCP parameter, change the value of a field and click **Update**. Table 20-1 describes the fields in the TCP window and the related CLI command. For more information on the TCP window fields, click the **HELP** button in the window.

- From the Content Engine CLI, use the **tcp** global configuration command to modify the TCP parameters, as described in Table 20-1.

**Note**

By default, the Content Engine does not automatically send out keepalives. To configure a standalone Content Engine to send out TCP keepalives on an HTTP connection, you must enter the **http tcp-keepalive enable** global configuration command. For more information on this topic, see the Configuring Standalone Content Engines to Send out TCP Keepalives, page 7-67.

| Table 20-1 TCP CLI Configuration Parameters |
|---|---|---|
| **Window Fields** | **TCP CLI Commands** | **Descriptions** |
| Send Buffer | **tcp server-send-buffer kbytes** | Server send buffer size in kilobytes (that is, the TCP outgoing window size [1–512 KB]). The default is 8 KB. |
### Table 20-1  TCP CLI Configuration Parameters (continued)

<table>
<thead>
<tr>
<th>Window Fields</th>
<th>TCP CLI Commands</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>To Client</td>
<td><code>tcp client-send-buffer kbytes</code></td>
<td>Client send buffer size in kilobytes (that is, the TCP outgoing window size [1–512 KB]). The default is 32 KB.</td>
</tr>
<tr>
<td><strong>Receive Buffer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To Server</td>
<td><code>tcp server-receive-buffer kbytes</code></td>
<td>Server receive buffer size in kilobytes (that is, the TCP incoming window size [1–512 KB]). The default is 32 KB.</td>
</tr>
<tr>
<td>To Client</td>
<td><code>tcp client-receive-buffer kbytes</code></td>
<td>Client receive buffer size in kilobytes (that is, the TCP incoming window size [1–512 KB]). The default is 8 KB.</td>
</tr>
<tr>
<td><strong>R/W Timeout</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To Server</td>
<td><code>tcp server-rw-timeout seconds</code></td>
<td>Interval after which the Content Engine times out trying to read or write to the network (1–3600). The default is 120 seconds.</td>
</tr>
<tr>
<td>To Client</td>
<td><code>tcp client-rw-timeout seconds</code></td>
<td>Interval after which the Content Engine times out trying to read or write to the network (1–3600). The default is 120 seconds.</td>
</tr>
<tr>
<td><strong>Keepalive</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td><code>tcp keepalive-timeout seconds</code></td>
<td>Length of time that the Content Engine keeps a connection open before disconnecting.</td>
</tr>
<tr>
<td>Interval</td>
<td><code>tcp keepalive-probe-interval seconds</code></td>
<td>Length of time that the Content Engine keeps an idle connection open (1–3600 seconds). The default is 300 seconds.</td>
</tr>
<tr>
<td>Count</td>
<td><code>tcp keepalive-probe-cnt count</code></td>
<td>Number of times the Content Engine retries a connection (1–10 attempts). The default is 4 attempts.</td>
</tr>
<tr>
<td>Congestion Window base value</td>
<td><code>tcp cwnd-base segments</code></td>
<td>Initial congestion window value (1–10 segments). The default is 2 segments.</td>
</tr>
<tr>
<td>Initial Slow Start Threshold value</td>
<td><code>tcp init-ss-threshold value</code></td>
<td>Threshold for slow start (2–10 segments). The default is 2 segments.</td>
</tr>
<tr>
<td>Retransmit Timer Increment factor</td>
<td><code>tcp increase-xmit-timer-value value</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Maximum Segment Size</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To Server</td>
<td><code>tcp server-mss maxsegsize</code></td>
<td>Maximum packet size sent to servers. The default is 1460 bytes.</td>
</tr>
<tr>
<td>To Client</td>
<td><code>tcp client-mss maxsegsize</code></td>
<td>Maximum packet size sent to clients. The default is 1432 bytes</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satellite</td>
<td><code>tcp server-satellite</code></td>
<td>Server and client TCP compliance with RFC 1323. See the “TCP-Over-Satellite Extensions” section on page 20-6.</td>
</tr>
<tr>
<td>Type of Service</td>
<td><code>type-of-service enable</code></td>
<td>TCP Type of Service. The default is disabled.</td>
</tr>
<tr>
<td>Ecnc</td>
<td><code>ecn enable</code></td>
<td>TCP explicit congestion notification.</td>
</tr>
<tr>
<td>TCP memory limits</td>
<td><code>tcp memory-limit</code></td>
<td>Configure TCP memory limits. See the next section, “Configuring TCP Memory Limits.”</td>
</tr>
</tbody>
</table>
Configuring TCP Memory Limits

In the ACNS 5.3.3 software and later releases, you can also use the CLI to configure TCP memory limits on a standalone Content Engine. The TCP memory limit settings allow you to control the amount of memory that can be used by the TCP subsystem’s send and receive buffers.

**Caution**

Do not modify the default values unless you know what you are doing. The default values are device dependent and have been chosen after extensive testing. They should not be changed under normal conditions. Increasing these values can result in the TCP subsystem using more memory, which might cause the system to become responsive. Decreasing these values can result in increased response times and lower performance.

For Content Engines that are registered with a Content Distribution Manager, you can also configure TCP memory limits through the Content Distribution Manager GUI. For information about how to configure TCP memory limits for Content Engines that are registered with a Content Distribution Manager, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*.

To configure the TCP memory limit settings for a standalone Content Engine, use the `tcp memory-limit` global configuration command.

Table 20-2 lists the CLI command options that were added in the ACNS 5.3.3 software release.

**Table 20-2  tcp memory-limit CLI Command Options**

<table>
<thead>
<tr>
<th>CLI Command Options</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>low-water-mark</td>
<td>Specifies the TCP limit low-water mark. This value specifies the lower limit (in MB) of the memory pressure mode, below which TCP enters into the normal memory allocation mode. The range is 4–600.</td>
</tr>
<tr>
<td>high-water-mark-pressure</td>
<td>Specifies the TCP memory limit high-water mark-pressure. This value specifies the upper limit (in MB) of the normal memory allocation mode, beyond which TCP enters into the memory pressure mode. The range is 5–610.</td>
</tr>
<tr>
<td>high-water-mark-absolute</td>
<td>Specifies the TCP memory limit high-water mark-absolute. This value specifies the absolute limit (in MB) on TCP memory usage. The range is 6–620.</td>
</tr>
</tbody>
</table>

In this example, the low-water mark is set to 4 MB and the high-water-mark-pressure is set to 5 MB:

```
ContentEngine(config#) tcp memory-limit low-water-mark 4 high-water-mark-pressure 5
```
Table 20-3 describes the default values for each command parameter, which are based on the total amount of memory for the device.

**Table 20-3  Default TCP Memory Limit Settings**

<table>
<thead>
<tr>
<th>Total System Memory</th>
<th>Low</th>
<th>Pressure</th>
<th>Absolute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB, 2 GB, or 4 GB</td>
<td>360 MB</td>
<td>380 MB</td>
<td>400 MB</td>
</tr>
<tr>
<td>512 MB</td>
<td>180 MB</td>
<td>190 MB</td>
<td>200 MB</td>
</tr>
<tr>
<td>256 MB</td>
<td>25 MB</td>
<td>28 MB</td>
<td>30 MB</td>
</tr>
</tbody>
</table>

The following conditions must be satisfied whenever these default values are changed:

- The low water mark must be a number that is less than the high water mark pressure setting.
- The high water mark pressure must be a number that is less than the high water mark absolute setting:
  
  \[
  \text{low-water-mark} < \text{high-water-mark-pressure} < \text{high-water-mark-absolute}
  \]

**Displaying TCP Configuration Information**

To display current TCP configuration information, use the `show tcp` EXEC command. The default 8 KB incoming window size for the client buffer is used here:

```
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
```

In this example, the `tcp client-receive-buffer` global configuration command is used to change the TCP incoming window size to 100 KB:

```
ContentEngine(config)# tcp client-receive-buffer 100
```
You can now verify the configuration change with the `show tcp` EXEC command.

```text
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
TCP memory_limit - Low water mark: 25 MB, High water mark (pressure): 28 MB,
High water mark (absolute): 30 MB
```

**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, which affects performance.

To set the TCP connection so that it complies with RFC 1323, use the `tcp server-satellite` and `tcp client-satellite` global configuration commands.
PART 5

Monitoring and Troubleshooting
Monitoring Standalone Content Engines and Transactions

This chapter describes how to monitor locally managed deployments (standalone Content Engines). This chapter contains the following sections:

- Monitoring Standalone Content Engines, page 21-2
- Monitoring Critical Disk Drives on Standalone Content Engines, page 21-17
- System Logging with Standalone Content Engines, page 21-21
- Monitoring Transactions with Standalone Content Engines, page 21-27
- Monitoring the Performance of Specific URLs, page 21-52

Note

In the ACNS 5.3.1 software and later releases, you can use the Secure File Transfer Protocol (SFTP) to connect to a Content Engine and securely retrieve log files from it.

For complete syntax and usage information for the CLI command used in this chapter, see the Cisco ACNS Software Command Reference, Release 5.5 publication. For information about monitoring a Content Router, Content Distribution Manager, or a Content Engine that is registered with a Content Distribution Manager (as opposed to standalone Content Engines that are not registered with a Content Distribution Manager), see the Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.
Monitoring Standalone Content Engines

It is important that you monitor your Content Engines in order to gauge their performance and to identify any signs that you need to tune their configurations or deploy additional Content Engines. This section describes how to use the Simple Network Management Protocol (SNMP) and the ACNS software alarms to monitor standalone Content Engines. Several tools are available to monitor the performance of standalone Content Engines that are running the ACNS 5.2.1 software and later releases. This set of tools includes the Cisco Discovery Protocol (CDP), SNMP, and the ACNS software alarms. For more information, see the following sections:

- Monitoring Standalone Content Engines with the Cisco Discovery Protocol, page 21-2
- Monitoring Standalone Content Engines with SNMP, page 21-2
- Monitoring Standalone Content Engines with the ACNS Software Alarms, page 21-11

Monitoring Standalone Content Engines with the 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 other devices within the network. 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, CiscoWorks2000 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 on a standalone Content Engine with a single CLI command:

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

Monitoring Standalone Content Engines with SNMP

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 SNMP 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.
Each Content Engine that is running the ACNS 5.x software has an SNMP agent that is responsible for gathering information about the Content Engine’s device configuration and activity. Before you can access this SNMP information, you must have deployed an SNMP management application on a management station. This SNMP management station is referred to as the SNMP host because it uses SNMP to send the device agent an SNMP Get request to obtain information from the Content Engine.

The SNMP management station and the device agent (the SNMP agent on the Content Engine) use SNMP to communicate, as follows:

1. The SNMP management station (the SNMP host) uses SNMP to request information from the Content Engine.

2. After receiving these SNMP requests, the device agent on the Content Engine accesses a table that contains information about the individual device (the Content Engine). This table, or database, is called a Management Information Base (MIB).

   Note: The SNMP agent on the Content Engine only initiates communication with the SNMP host under unusual conditions; it will initiate communication when it has a trap it needs to send to the host. For more information on this topic, see the “Configuring Standalone Content Engines to Send SNMP Traps” section on page 21-8.

3. After locating the specified information in the MIB, the device agent uses SNMP to send the information to the SNMP management station.

Figure 21-1 illustrates these SNMP operations when the Content Engine has been standalone.

Figure 21-1    SNMP Components with a Standalone ACNS Content Engine
Understanding the Different Versions of SNMP

The ACNS 5.x software supports the following versions of SNMP:

- **Version 1 (SNMPv1)**—This is the initial implementation of SNMP. See the 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.

SNMP Security Models and Security Levels

SNMPv1 and SNMPv2c do not have any security (that is, authentication or privacy) mechanisms to keep SNMP packet traffic 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. The SNMP agent in the ACNS 5.x software supports SNMPv3 as well as 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 security models as well as 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 an SNMP packet is handled. Three security models are available: SNMPv1, SNMPv2c, and SNMPv3.

Table 21-1 describes the combinations of security models and security levels.

<table>
<thead>
<tr>
<th>Model</th>
<th>Level</th>
<th>Authentication</th>
<th>Encryption</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>noAuthNoPriv</td>
<td>Community string</td>
<td>No</td>
<td>Uses a community string match for user authentication.</td>
</tr>
<tr>
<td>v2c</td>
<td>noAuthNoPriv</td>
<td>Community string</td>
<td>No</td>
<td>Uses a community string match for user authentication.</td>
</tr>
<tr>
<td>v3</td>
<td>noAuthNoPriv</td>
<td>Username</td>
<td>No</td>
<td>Uses a username match for user authentication.</td>
</tr>
<tr>
<td>v3</td>
<td>AuthNoPriv</td>
<td>Message Digest 5 (MD5)</td>
<td>No</td>
<td>Provides authentication based on the Hash-Based Message Authentication Code (HMAC)-MD5 or HMAC-SHA algorithms.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or Secure Hash Algorithm (SHA)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td>AuthPriv</td>
<td>MD5 or SHA</td>
<td>Yes</td>
<td>Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides Data Encryption Standard (DES) 56-bit encryption (packet authentication) based on the cipher block chaining (CBC)-DES (DES-56) standard.</td>
</tr>
</tbody>
</table>
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 be performed on the packet)

Using SNMPv3, users can securely collect management information from their SNMP agents without worrying 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.

### Supported MIBs

The ACNS 5.x software implementation of SNMP supports the following MIBs:

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

**Note**

In the ACNS 5.2.1 software and later releases, the CISCO-CONTENT-ENGINE-MIB supports streaming WMT (MMS and MMS-over-HTTP), RealProxy, and Cisco Streaming Engine statistics. Standalone Content Engines support WMT and RealProxy.

Cisco Streaming Engine is only supported on Content Engines that are registered with a Content Distribution Manager; Cisco Streaming Engine is not supported on standalone Content Engines. In the ACNS 5.5 software, the CISCO-CONTENT-ENGINE-MIB supports streaming WMT for only MMS-over-HTTP.

**Note**

In the ACNS 5.3.1 software release, the CISCO-CONTENT-ENGINE-MIB was modified to add support for WMT RTSP streaming for Windows Media 9 clients and servers (that is, Windows Media 9 players and Windows Media 9 servers).

In the ACNS 5.2.1 software and later releases, there are six generic alarm traps in the CISCO-CONTENT-ENGINE-MIB for SNMP and Node Health Manager integration. For a list of these six generic alarm traps, see Table 21-5.

In the ACNS 5.1.1 software and later releases, you can use IP access control lists (ACLs) to control SNMP access on a Content Engine. For more information about defining IP ACLs, see Chapter 19, “Creating and Managing IP Access Control Lists for Standalone Content Engines.”
Download MIB Files to Standalone Content Engines

You can download the MIB files for all of the MIBs that are supported by a standalone Content Engine that is running the ACNS 5.x software, from the following Cisco FTP site:

The MIB objects that are defined in each MIB are described in the MIB files at the above FTP site are self explanatory.

Enabling the SNMP Agent on Standalone Content Engines

By default, the SNMP agent on a standalone Content Engine is disabled and an SNMP community string is not defined. The SNMP community string is used as a password for authentication when accessing the SNMP agent on the standalone Content Engine. In order to be authenticated, the Community Name field of any SNMP message sent to the standalone Content Engine must match the SNMP community string defined on the standalone Content Engine.

The SNMP agent on the standalone Content Engine is enabled when an SNMP community string is defined on the Content Engine. You can use the Content Engine GUI or CLI to define an SNMP community string and enable the SNMP agent, as follows:

- From the Content Engine GUI, choose System > SNMP. In the displayed SNMP window, scroll down to the Community field and enter an SNMP community string. Click Update.
- From the Content Engine CLI, use the `snmp-server community` command:

```
ContentEngine(config)# snmp-server community comaccess
```

If the SNMPv3 protocol is going to be used for SNMP requests, the next step is to define an SNMP user account that can be used to access a standalone Content Engine through SNMP. For more information on how to create an SNMPv3 user account on a standalone Content Engine, see the “Defining SNMP Users for Standalone Content Engines” section on page 21-6.

Defining SNMP Users for Standalone Content Engines

When defining SNMP users for standalone Content Engines, remember the following important points:

- If the SNMPv3 protocol is going to be used for SNMP requests, you must define at least one SNMPv3 user account on the standalone Content Engine in order for the Content Engine to be accessed through SNMP.
- A group defined with the SNMPv1 or SNMPv2c security model should not be associated with SNMP users; they should only be associated with the community strings.
Defining SNMPv3 Users

You can use either the Content Engine GUI or the CLI to define SNMPv3 user accounts on standalone Content Engines.

To use the Content Engine GUI to configure an SNMPv3 user account on a standalone Content Engine, follow these steps:

**Step 1**
From the Content Engine GUI, choose **System > SNMP**.
The SNMP window for configuring SNMPv1 or SNMPv2 appears.

**Step 2**
Scroll down to the bottom of the SNMP window. At the bottom of the window, click the SNMPv3 Configuration **Click here** link.
The SNMP window for configuring SNMPv3 settings (including SNMPv3 user accounts) appears.

**Step 3**
Scroll down to the SNMPv3 User configuration section of the SNMP v3 window. Use the SNMPv3 user configuration fields and drop-down lists to define new SNMPv3 user accounts on this Content Engine.

a. In the Name field, enter the name of the SNMP user. Use letters, numbers, dashes, and underscores, but no blanks. This is the name of the user on the SNMP host who wants to communicate with the SNMP agent on the Content Engine.

b. In the Group field, enter the name of the group to which the SNMP user belongs.

c. In the Remote SnmpID field, enter the globally unique identifier for a remote SNMP entity (for example, the SNMP network management station) for at least one of the SNMP users.

Tip
In order to send an SNMPv3 inform message, at least one SNMPv3 user with a remote SNMP ID option must be configured on the Content Engine. The SNMP ID is entered in octet string form. For example, if the IP address of a remote SNMP entity is 192.147.142.129, then the octet string would be 00:00:63:00:00:a1:c0:93:8e:81.

d. From the **Auth-Algorithm** drop-down list, choose the algorithm used to authenticate the SNMP user (**md5**, **sha**, or **no_auth**). By default, **no_auth** for no authentication is chosen.
   - Choose **md5** for the HMAC-MD5-96 authentication level.
   - Choose **sha** for the HMAC-SHA-96 authentication level.

e. In the optional Auth-Password field, enter the HMAC MD5 user authentication password. This field is only applicable if you chose **md5** as the authentication type.

f. In the optional Priv-Password field, enter the HMAC MD5 user private password. This field is only applicable if you chose **md5** as the authentication type. This is a string that enables the SNMP agent to receive packets from the host.

**Step 4**
Click **Update** to add the new SNMPv3 user account.
The new user account that you just created is listed in the SNMPv3 window.

**Step 5**
Continue to add more SNMPv3 user accounts. To remove an existing SNMPv3 user account, click the **Delete** check box next to the account that you want to remove.

**Step 6**
Click **Update** again to save your changes to the SNMPv3 user accounts.
To use the Content Engine CLI to define an SNMPv3 user account on a standalone Content Engine (the SNMP server), use the `snmp-server user` global configuration command. Use the `no` form of this command to remove SNMP access.

```
snmp-server user name group [auth {md5 password [priv password] | sha password [priv password]} | remote octetstring [auth {md5 password [priv password] | sha password [priv password]}]]
```

Table 21-2 describes the parameters for the `snmp-server user` command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the SNMP user.</td>
</tr>
<tr>
<td>group</td>
<td>Group to which the SNMP user belongs.</td>
</tr>
<tr>
<td>auth</td>
<td>(Optional) Configures user authentication parameters.</td>
</tr>
<tr>
<td>md5</td>
<td>Configures the HMAC MD5 authentication algorithm.</td>
</tr>
<tr>
<td>password</td>
<td>HMAC MD5 user authentication password.</td>
</tr>
<tr>
<td>priv</td>
<td>(Optional) Configures the authentication parameters for the packet.</td>
</tr>
<tr>
<td>sha</td>
<td>Configures the HMAC SHA authentication algorithm.</td>
</tr>
<tr>
<td>password</td>
<td>HMAC SHA authentication password.</td>
</tr>
<tr>
<td>remote</td>
<td>(Optional) Specifies engine identity octet string.</td>
</tr>
<tr>
<td>octetstring</td>
<td>Engine identity octet string.</td>
</tr>
</tbody>
</table>

In the following example, an SNMPv3 user account is created on the Content Engine. The SNMPv3 user is named `acme` and belongs to the group named `admin`. Because this SNMP user account has been set up with no authentication password, the SNMP agent on the Content Engine will not perform authentication on SNMP requests from this user.

```
ContentEngine(config)# snmp-server user acme admin
```

**Configuring Standalone Content Engines to Send SNMP Traps**

You can use either the Content Engine GUI or the CLI to configure standalone Content Engines to send SNMP traps.

From the Content Engine GUI, choose **System > SNMP**. The SNMP window appears. From the SNMP window, do one of the following:

- To configure the Content Engine SNMPv1 or SNMPv2 agent to send SNMP traps to a specific SNMP host, enter information in the appropriate fields of this window, and click **UPDATE**.
  
  For example, you must define the SNMP trap host by specifying the hostname or IP address of the SNMP trap host that will be sent in the SNMP trap messages from the Content Engine.

- To configure the Content Engine’s SNMP v3 agent to send SNMP traps to a specific SNMP host, scroll down to the bottom of the SNMP window. Click the SNMPv3 Configuration **Click here** link. The SNMP window for configuring an SNMPv3 agent on the Content Engine appears. Use the SNMPv3 window to configure SNMP traps and SNMP v3 user accounts for this Content Engine.
For more information about configuring SNMPv3 user accounts, see the “Defining SNMPv3 Users” section on page 21-7. For information about the fields in the SNMP windows, click the HELP button in the window.

When using the Content Engine CLI to configure a standalone Content Engine to send SNMP traps, remember the following important points:

- For an SNMP host to receive a trap, both the snmp-server enable traps command and the snmp-server host command for that host must be configured. In addition, SNMP must be enabled with the snmp-server community command.

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

To use the Content Engine CLI to configure SNMP traps on a standalone Content Engine, follow these steps:

**Step 1** Choose one of the security model groups (SNMPv1, SNMPv2c, or SNMPv3) by using the `snmp-server group name` global configuration command.

```
snmp-server group name {v1 [notify name] [read name] [write name] | v2c [notify name] [read name] [write name] | v3 [auth [notify name] [read name] [write name] | noauth [notify name] [read name] [write name] | priv [notify name] [read name] [write name]}
```

where:

- `name` Name of group.
- `v1` Specifies the group using the Version 1 Security Model.
- `notify` (Optional) Specifies a notify view for the group.
- `name` Notify view name.
- `read` (Optional) Specifies a read view for the group.
- `name` Read view name.
- `write` (Optional) Specifies a write view for the group.
- `name` Write view name.
- `v2c` Specifies the group using the Version 2c Security Model.
- `v3` Specifies the group using the User Security Model (SNMPv3).
- `auth` Specifies the group using the AuthNoPriv Security Level.
- `noauth` Specifies the group using the noAuthNoPriv Security Level.
- `priv` Specifies the group using the AuthPriv Security Level.

**Step 2** Enable all SNMP traps on the Content Engine.

```
ContentEngine(config)# snmp-server enable traps
```

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.

**Step 3** Specify which host or hosts receive SNMP traps from the Content Engine.

The following example shows how to configure the Content Engine to send all SNMP traps to the host 172.31.2.160 using the community string public:

```
ContentEngine(config)# snmp-server host 172.31.2.160 public
```
**Note**

To send SNMP traps, you must configure at least one SNMP trap host. In the ACNS 5.1 software and earlier releases, you could only configure up to four SNMP hosts. In the ACNS 5.2.1 software and later releases, you can configure up to eight SNMP hosts on a Content Engine.

**Step 4**

Enable the SNMP agent on the Content Engine, and assign a community string as a password for authentication when you access the SNMP agent on the Content Engine.

The following example shows how to specify comaccess as the password:

```
ContentEngine(config)# snmp-server community comaccess
```

**Tip**

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.

The **snmp-server community string** global configuration command provides view-based access control for SNMPv1, SNMPv2c, and SNMPv3 but also continues to provide backward compatibility between different versions.

In the ACNS 5.x software prior to the ACNS 5.1 software release, the **snmp-server community string** global configuration 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** EXEC command, but are created during initialization of the SNMP agent.

---

**Disabling the SNMP Agent on Standalone Content Engines**

To disable the SNMP agent on standalone Content Engines, enter the **no snmp-server** global configuration command:

```
ContentEngine(config)# no snmp-server
```

To disable the SNMP agent and to remove the previously defined community string, enter the **no snmp-server community** global configuration command:

```
ContentEngine(config)# no snmp-server community
```
Disabling SNMP Traps on Standalone Content Engines

To disable all SNMP traps on standalone Content Engines, enter the `no snmp-server enable traps` global configuration command:

```
ContentEngine(config)# no snmp-server enable traps
```

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

Monitoring Standalone Content Engines with the ACNS Software Alarms

Traditionally SNMP is used to report error conditions by generating SNMP traps. ACNS 5.x continues to use this monitoring method, as described in the “Monitoring Standalone Content Engines with SNMP” section on page 21-2.

In the ACNS 5.2.1 software and later releases, the Node Health Manager feature is supported. The Node Health Manager enables ACNS applications to raise alarms to draw attention to significant problems. The Node Health Manager, which is the data repository for such alarms, aggregates the health and alarm information for the applications, services (for example, the cache service) and resources (for example, disk drives) that are being monitored on the Content Engine. For example, this new feature gives you a way to determine if a monitored application (for example, the HTTP proxy caching service) is alive on the Content Engine. These alarms are referred to as the ACNS software alarms.

In the ACNS 5.2.1 software and later releases, the following Content Engine applications can generate an ACNS software alarm:

- Node Health Manager (Alarm overload condition and Node Manager aliveness)
- Node Manager for service failures (aliveness of monitored applications)
- System Monitor (sysmon) for disk failures

Alarms that have been raised on a Content Engine can be listed using the Content Engine CLI commands in Table 21-3. SNMP traps are sent all raised and cleared alarms. The type of SNMP trap sent varies by alarm.

**Note**

If the Content Engine is registered with a Content Distribution Manager, the Node Health Manager also sends a notification about the alarm to the Content Distribution Manager. For more information on this topic, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.*

In the ACNS 5.2.1 software release, several CLI commands were added to allow you to systematically identify the source of an ACNS software alarm (the cause of the problem). (See Table 21-3.) You can use these CLI commands to identify the source of a problem without searching through numerous ACNS software logs.
Chapter 21  Monitoring Standalone Content Engines and Transactions

Monitoring Standalone Content Engines

Note
On standalone Content Engines, information about the ACNS software alarms is available through the Content Engine CLI and also through SNMP. See Table 21-3 for a list of the CLI commands that you can use to access this alarm information for a standalone Content Engine.

Alarm Severity Levels

There are three levels of ACNS software alarms. (See Table 21-4.)

Table 21-3  List of Show Alarms CLI Commands

<table>
<thead>
<tr>
<th>CLI Command</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>show alarm</td>
<td>Displays a list of all currently raised ACNS software alarms (critical, major, and minor alarms) on the Content Engine.</td>
<td>See the “Displaying Information About ACNS Software Alarms” section on page 21-15.</td>
</tr>
<tr>
<td>show alarm critical</td>
<td>Displays a list of only currently raised ACNS software critical alarms on the Content Engine.</td>
<td>See the “Displaying Information About ACNS Software Alarms” section on page 21-15.</td>
</tr>
<tr>
<td>show alarm major</td>
<td>Displays a list of only currently raised ACNS software major alarms on the Content Engine.</td>
<td>See the “Displaying Information About ACNS Software Alarms” section on page 21-15.</td>
</tr>
<tr>
<td>show alarm minor</td>
<td>Displays a list of the currently raised ACNS software minor alarms on the Content Engine.</td>
<td>See the “Displaying Information About ACNS Software Alarms” section on page 21-15.</td>
</tr>
<tr>
<td>show alarm detail</td>
<td>Displays detailed information about the currently raised ACNS software alarms.</td>
<td>See the “Displaying Details About ACNS Software Alarms” section on page 21-15.</td>
</tr>
<tr>
<td>show alarms history</td>
<td>Displays a history of ACNS software alarms that have been raised and cleared on the Content Engine. The CLI retains the last 100 alarm raise and clear events only.</td>
<td>See the “Displaying the History of ACNS Software Alarms” section on page 21-15.</td>
</tr>
<tr>
<td>show alarms status</td>
<td>Displays the counts for the currently raised ACNS software alarms on the Content Engine. Also lists the alarm overload state and the alarm overload settings.</td>
<td>See the “Displaying the Status of ACNS Software Alarms” section on page 21-17.</td>
</tr>
</tbody>
</table>

Table 21-4  Levels of Alarm Severity for ACNS Software Alarms

<table>
<thead>
<tr>
<th>Alarm Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>Alarms that affect the existing traffic through the Content Engine, and are considered fatal (Content Engine cannot recover and continue to process traffic).</td>
</tr>
<tr>
<td>Major</td>
<td>Alarms that indicate a major service (for example, the cache service) is damaged or lost. Urgent action is necessary to recover this service. However, other node components are fully functional and the existing service should be minimally impacted.</td>
</tr>
<tr>
<td>Minor</td>
<td>Alarms that indicate a condition that will not affect a service (a non-service-affecting condition) occurred but that corrective action is required in order to prevent a serious fault from occurring.</td>
</tr>
</tbody>
</table>
The output of the `show alarms history` EXEC command includes the severity of the ACNS software alarm:

```
ContentEngine# show alarms history
```

```
<table>
<thead>
<tr>
<th>Op</th>
<th>Sev</th>
<th>Alarm ID</th>
<th>Module/Submodule</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>cache</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>cache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>rpc_httpd</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>rpc_httpd</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>rpc_httpd</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>rpc_httpd</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>cache</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>cache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>cache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>C</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>rtspg</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>cache</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>mediacache</td>
</tr>
<tr>
<td>R</td>
<td>Mi</td>
<td>servicedead</td>
<td>nodemgr</td>
<td>rtspg</td>
</tr>
</tbody>
</table>

Op - Operation: R-Raised, C-Cleared
Sev - Severity: Cr-Critical, Ma-Major, Mi-Minor

**Alarm Overload**

In the ACNS 5.2.1 software and later releases, Content Engines can track the rate of incoming alarms from the Node Health Manager. If the rate of incoming alarms exceeds the high-water mark (HWM), then the Content Engine enters an alarm overload state. When a standalone Content Engine is in an alarm overload state, then the following occurs:

- SNMP traps for subsequent alarm raise and clear operations are suspended. The trap for the raise alarm-overload alarm and the clear alarm-overload alarm are sent; however traps related to alarm operations between the raise alarm-overload alarm and the clear alarm-overload alarm operations are suspended.
- Alarm overload raise and clear notifications are not blocked.
- The Content Engine remains in an alarm overload state if the rate of incoming alarms decreases to the point that the alarm rate is less than the low-water mark (LWM).

**Checking for Application Liveness**

The Node Manager tracks the liveness of applications that it creates on the Content Engine (for example, the HTTP cache application, the WMT streaming application, and the RTSP gateway (RTSPG) streaming application). When the Node Manager detects the termination of a spawned application, it raises an alarm. An application is considered dead when the Node Manager does not receive keepalives from the application.

When an application dies, the Node Manager raises a servicedied alarm to report the condition, and then it restarts the service. If the service continues to run for a short time (typically 10 seconds), the servicedied alarm is cleared.
If the application dies again after restarting, the servicedied alarm continues to be raised and the Node Manager attempts to restart it. Restarts are done typically a maximum of 10 times by the Node Manager. After that, the Node Manager raises a svcdisabled alarm for the service, clears the 'servicedied' alarm, and it stops restarting the service.

To restart the service, you must unconfigure and configure the features (for example, in the case of the NTP service, enter the `no ntp server hostname | IP address` global configuration command to unconfigure the NTP service, and then enter the `ntp server hostname | IP address` global configuration command to reconfigure the NTP service.

### Configuring SNMP Alarm Traps on Standalone Content Engines

You can configure a Content Engine to generate an SNMP trap for a specific alarm condition. You can configure the generation of SNMP alarm traps on standalone Content Engines based on the following:

- The severity of the alarm (critical, major, or minor)
- The action (the alarm is raised or cleared)

In the ACNS 5.2.1 software release, the following six generic alarm traps were added to the CISCO-CONTENT-ENGINE-MIB (the CCE MIB). See Table 21-5.

#### Table 21-5  Generic Alarm Traps

<table>
<thead>
<tr>
<th>Name of Alarm Trap</th>
<th>Severity</th>
<th>Action</th>
<th>CLI Command To Enable Alarm Trap</th>
</tr>
</thead>
<tbody>
<tr>
<td>cceAlarmCriticalRaised</td>
<td>Critical</td>
<td>Raised</td>
<td><code>snmp-server enable traps alarm raise-critical</code></td>
</tr>
<tr>
<td>cceAlarmCriticalCleared</td>
<td>Critical</td>
<td>Cleared</td>
<td><code>snmp-server enable traps alarm clear-critical</code></td>
</tr>
<tr>
<td>cceAlarmMajorRaised</td>
<td>Major</td>
<td>Raised</td>
<td><code>snmp-server enable traps alarm raise-major</code></td>
</tr>
<tr>
<td>cceAlarmMajorCleared</td>
<td>Major</td>
<td>Cleared</td>
<td><code>snmp-server enable traps alarm clear-major</code></td>
</tr>
<tr>
<td>cceAlarmMinorRaised</td>
<td>Minor</td>
<td>Raised</td>
<td><code>snmp-server enable traps alarm raise-minor</code></td>
</tr>
<tr>
<td>cceAlarmMinorCleared</td>
<td>Minor</td>
<td>Cleared</td>
<td><code>snmp-server enable traps alarm clear-minor</code></td>
</tr>
</tbody>
</table>

By default, these six general alarm traps are disabled.

These six general alarm traps provide SNMP and Node Health Manager integration. Each of these six alarm traps can be enabled or disabled through the Content Engine CLI. In the ACNS 5.2.1 software and later releases, the `snmp-server enable traps global configuration command includes an alarm` option.

```
ContentEngine(config)# snmp-server enable traps alarm ?
clear-critical  Enable clear-critical alarm trap
clear-major     Enable clear-major alarm trap
clear-minor     Enable clear-minor alarm trap
raise-critical  Enable raise-critical alarm trap
raise-major     Enable raise-major alarm trap
raise-minor     Enable raise-minor alarm trap
```

In the following example, the Content Engine (the SNMP server) is configured to generate an SNMP trap if a critical alarm is cleared:

```
ContentEngine(config)# snmp-server enable traps alarm clear-critical
```
Displaying Information About ACNS Software Alarms

To display information about all of the currently raised critical, major, and minor alarms for a standalone Content Engine, enter the `show alarm` EXEC command. If there are no alarms currently raised on the Content Engine, the output indicates “None.” The following shows a sample output:

```
ContentEngine# show alarm

Critical Alarms: ----------------
None

Major Alarms: --------------
None

Minor Alarms: -------------
None
```

You can also display information for only a specific level of ACNS software alarms that are currently raised on the Content Engine, as follows:

- To display information about only the critical alarms, enter the `show alarm critical` EXEC command.
- To display information about only the major alarms, enter the `show alarm major` EXEC command.
- To display information about only the minor alarms, enter the `show alarm minor` EXEC command.

**Note**
For a description of the various severity levels for alarms (critical, major, and minor), see Table 21-4.

Displaying Details About ACNS Software Alarms

To display details about the currently raised SNMP alarms, enter the `show alarm detail` EXEC command. This command allows you to identify more information about a particular alarm.

Displaying the History of ACNS Software Alarms

To display a history of ACNS software alarms that have been raised and cleared on a standalone Content Engine, enter the `show alarms history` EXEC command:

```
ContentEngine# show alarms history

Op Sev Alarm ID       Module/Submodule       Instance
------------- ----------------- ---------------
1 C  Mi servicedead   nodemgr              mediacache
2 C  Mi servicedead   nodemgr              cache
3 R  Mi servicedead   nodemgr              mediacache
4 R  Mi servicedead   nodemgr              cache
5 C  Mi servicedead   nodemgr              rpc_httpd
6 R  Mi servicedead   nodemgr              rpc_httpd
7 C  Mi servicedead   nodemgr              rpc_httpd
8 R  Mi servicedead   nodemgr              rpc_httpd
9 C  Mi servicedead   nodemgr              mediacache
10 C Mi servicedead   nodemgr              cache
11 R Mi servicedead   nodemgr              mediacache
12 R Mi servicedead   nodemgr              cache
13 C Mi servicedead   nodemgr              cache
```
Chapter 21 Monitoring Standalone Content Engines and Transactions

To display additional information about an alarm, enter the `show alarms history detail support` EXEC command:

```
Content Engine# show alarms history detail support
Op Sev Alarm ID             Module/Submodule     Instance
-- --- -------------------- -------------------- -------------------
1 C  Mi  servicedead          nodemgr              rtspg
    nodemgr: The rtspg service has died.
    /alm/min/nodemgr/-service_name-/servicedead:
        -service name- service has died.
    Explanation:
        The node manager found the specified service to be dead.
        Attempts will be made to restart this service.
    Action:
        Examine the syslog for messages relating to cause of service
        death. The alarm will be cleared if the service stays
        alive and does not restart in a short while.

2 R  Mi  servicedead          nodemgr              rtspg
    Jul 2 18:21:54.231 UTC, Processing Error Alarm, #000001, 2000:330004
    nodemgr: The rtspg service has died.
    /alm/min/nodemgr/-service_name-/servicedead:
        -service name- service has died.
    Explanation:
        The node manager found the specified service to be dead.
        Attempts will be made to restart this service.
    Action:
        Examine the syslog for messages relating to cause of service
        death. The alarm will be cleared if the service stays
        alive and does not restart in a short while.
```

Op - Operation: R-Raised, C-Cleared
Sev - Severity: Cr-Critical, Ma-Major, Mi-Minor
Displaying the Status of ACNS Software Alarms

To display the counts for all currently raised alarms on the Content Engine, enter the `show alarms status` EXEC command. The following sample output shows the number of currently raised ACNS software alarms. The output also includes information about the alarm overload settings (for example, if overload detection is currently enabled or disabled on the Content Engine).

```
ContentEngine# show alarms status
Critical Alarms : 0
Major Alarms : 0
Minor Alarms : 0

Overall Alarm Status : None
Device is NOT in alarm overload state.
Device enters alarm overload state @ 10 alarms/sec.
Device exits alarm overload state @ 1 alarms/sec.
Overload detection is DISABLED.
```

Monitoring Critical Disk Drives on Standalone Content Engines

To operate properly, the Content Engine depends on the following critical disk drives:

- The first disk drive that is referred to as “disk00.”
- The disk drive that contains the first sysfs (system file system) partition.
  
  The sysfs partition is used to store log files, including transaction logs, system logs (syslogs), and internal debugging logs. It can also be used to store image files and configuration files on a Content Engine.

**Note**

The term *critical drive* is defined as a disk drive that is either disk00 or a disk drive that contains the first sysfs partition. A critical drive can be different based on the Content Engine model. For example, with smaller, single disk drive Content Engines there is only one critical disk drive; with higher-end Content Engines that have more than one disk drive, there may be more than one critical disk drive on the Content Engine.

When a Content Engine is booted and a critical disk drive is not detected at system startup time, the ACNS system on the Content Engine runs at a degraded state. If one of the critical disk drives goes bad at run time, the applications might malfunction, suspend operations, or stop operating, or the ACNS system might suspend or stop operations. Consequently, it is important that the critical disk drives on a Content Engine are monitored and that disk drive errors are reported.

With an ACNS system, a disk device error is defined as any of the following events:

- A Small Computer Systems Interface (SCSI) or Integrated Drive Electronics (IDE) device error is printed by Linux kernel.
- A disk device access by an application (for example, an open(2), read(2), or write(2) system call) fails with an EIO error code.
- A disk device that existed at startup time is not accessible at run time.

In the ACNS 5.2.1 software and later releases, you can monitor Content Engine disk drives. Disk status is recorded in flash (non-volatile storage). When an error on a Content Engine disk device occurs, a message is written to the system log (syslog) if the sysfs partition is still intact, and an SNMP trap is generated if SNMP is configured on the Content Engine.
In addition to tracking the state of critical disk drives, you can define a disk device error-handling threshold on the Content Engine. If the number of disk device errors reaches the specified threshold, the corresponding disk device is automatically marked as bad. The ACNS system does not stop using the bad disk device immediately; it stops using the bad disk drive after the next reboot.

If the specified threshold is exceeded, the Content Engine either records this event or reboots. If the automatic reload feature is enabled and this threshold is exceeded, then the ACNS system automatically reboots the Content Engine. For more information about specifying this threshold, see the “Specifying the Disk Error-Handling Threshold” section on page 21-18.

**Note** You can also manually mark a disk drive as bad or good by using the `disk drive mark` EXEC command. For more information on this topic, see the “Manually Marking and Unmarking Content Engine Disk Drives” section on page 21-19.

In the ACNS 5.2.1 software release, support for remapping bad (but unused) sectors on a SCSI drive was added. In the ACNS 5.3.1 software and later releases, this capability includes IDE and Serial Advanced Technology Attachment [SATA] drives. For more information on this topic, see the *Cisco ACNS Software Upgrade and Maintenance Guide, Release 5.x.*

### Specifying the Disk Error-Handling Threshold

In the ACNS 5.2.1 software and later releases, you can configure a disk error-handling threshold. This threshold determines how many disk errors can be detected before the disk drive is automatically marked as bad. By default, this threshold is set to 10.

To change the default threshold, use the `disk error-handling threshold` global configuration command. Valid values are from 0 through 100. Specify 0 if you never want the disk drive to be marked as bad.

In the following example, five disk drive errors for a particular disk drive (for example, disk00) will be allowed before the disk drive is automatically marked as bad:

```
ContentEngine(config)# disk error-handling threshold 5
```

If the bad disk drive is a critical disk drive, and the automatic reload feature (`disk error-handling reload` command) is enabled, then the ACNS software marks the disk drive as bad and the Content Engine is automatically reloaded. After the Content Engine is reloaded, a syslog message and an SNMP trap are generated.

By default, the automatic reload feature is disabled on a Content Engine. To enable the automatic reload feature, use the `disk error-handling reload` global configuration command:

```
ContentEngine(config)# disk error-handling reload
```

To disable the automatic reload feature, enter the `no disk error-handling reload` global configuration command:

```
ContentEngine(config)# no disk error-handling reload
```
Manually Marking and Unmarking Content Engine Disk Drives

A disk drive on a Content Engine will remain in the Not used state until you manually unmark it as follows:

- To reset the disk state, use one of the following `disk add` EXEC commands on a standalone Content Engine. The disk state Not used is reset if you use any of these `disk add` commands.

  ```
  disk add diskname [sysfs {remaining | disk-space}] [cfs {remaining | disk-space}] [mediafs {remaining | disk-space}]
  ```

- To mark one or all disk drives manually as good (being used) or bad (will not be used after reload), use the `disk mark` EXEC command.

The following example shows how to mark disk03 as bad, reload the Content Engine, then unmark disk03 as a bad so that it can be used again:

**Step 1** Mark disk03 as bad.

Content Engine# `disk mark`?

```
WORD  Disk name (e.g. disk00, disk01,..)
```

Content Engine# `disk mark disk03`?

```
bad   Mark as bad disk drive, don't use it
good  Mark as good disk drive
```

Content Engine# `disk mark disk03 bad`

```
disk03 is marked as bad.
It will be not used after reload.
```

**Step 2** Verify that disk03 is marked as "*" because it was marked after the Content Engine was booted.

Content Engine# `show disks details`

```
(*) Disk drive won't be used after reload.
```

```
disk03: Normal        (h00 c00 i03 l00 - Int DAS)        70001MB( 68.4GB) (*)
FREE:                    70001MB( 68.4GB)
```

Notice that disk03 is reported as Normal (currently being used).

**Step 3** Reload the Content Engine by entering the `reload` EXEC command. When asked, press Enter to proceed with the reload.

Content Engine# `reload`

```
Proceed with reload?[confirm]
```

```
......
```

After the Content Engine is reloaded, disk03 that is marked as a bad disk drive will not be used.

**Step 4** Verify that disk03 is marked as not used.

Content Engine# `show disks details`

```
(*) Disk drive won't be used after reload.
```

```
disk03: Not used (*)
```

```
......
```

Disk03 is now shown as not used (*) because disk03 was detected as bad after the Content Engine was rebooted.

**Step 5** Unmark disk03 as bad by manually marking it as good.

Content Engine# `disk mark disk03 good`

```
disk03 is marked as good.
```
It will be used after reload.

**Step 6** Verify that disk03 is now marked as Not used.

Content Engine# show disks details

....... 
disk03: Not used 
....... 

---

**Proactively Monitoring Disk Health with SMART**

In the ACNS 5.3.1 software and later releases, you can proactively monitor the health of disks with Self Monitoring, Analysis, and Reporting Technology (SMART). SMART provides you with hard drive diagnostic information and information about impending disk failures.

SMART is supported by most disk vendors and is a standard method used to determine how healthy a disk is. SMART attributes include several read-only attributes (for example, the power on hours attribute, the load and unload count attribute) that provide the ACNS software with information regarding the operating and environmental conditions that may indicate an impending disk failure.

SMART support is vendor dependent; each disk vendor has a different set of supported SMART attributes. In the following sample output, the `show disk SMART-info` EXEC command was entered on two different Content Engines (Content Engine A and Content Engine B). These two Content Engines contain hard disks that were manufactured by different vendors.

ContentEngineA# show disks SMART-info

```bash
=== disk00 ===
Device: IBM IC35L036UCD210-0 Version: SSBS
Serial number: 22222222
Device type: disk
Transport protocol: Fibre channel (FCP-2)
Local Time is: Sun Jan 2 03:14:16 2005 Etc
Device supports SMART and is Enabled
Temperature Warning Disabled or Not Supported
SMART Health Status: OK

=== disk01 ===
disk01: Not present
```

ContentEngineB# show disk SMART-info

```bash
Disk 01:
========
Device Model: HITACHI_DK23BA-20
Serial Number: 111111
Firmware Version: 00E0A0D2
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
SMART overall-health self-assessment test result: PASSED
Vendor Specific SMART Attributes with Thresholds:

<table>
<thead>
<tr>
<th>#</th>
<th>ATTRIBUTE_NAME</th>
<th>FLAG</th>
<th>VALUE</th>
<th>WORST</th>
<th>THRESH</th>
<th>TYPE</th>
<th>WHEN_FAILED</th>
<th>RAW_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Raw_Read_Error_Rate</td>
<td>0x000d</td>
<td>100</td>
<td>081</td>
<td>050</td>
<td>Pre-fail</td>
<td>-</td>
<td>677</td>
</tr>
<tr>
<td>3</td>
<td>Spin_Up_Time</td>
<td>0x0007</td>
<td>100</td>
<td>100</td>
<td>050</td>
<td>Pre-fail</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Start_Stop_Count</td>
<td>0x0032</td>
<td>100</td>
<td>100</td>
<td>050</td>
<td>Old_age</td>
<td>-</td>
<td>249</td>
</tr>
<tr>
<td>5</td>
<td>Reallocated_Sector_Ct</td>
<td>0x0033</td>
<td>099</td>
<td>099</td>
<td>010</td>
<td>Pre-fail</td>
<td>-</td>
<td>30</td>
</tr>
</tbody>
</table>
```

---
To display more detailed information, enter the `show disk SMART-info details` EXEC command. The output from the `show disk SMART-info` and the `show disk SMART-info details` commands will differ based on the disk vendor and the type of drive technology (Integrated Drive Electronics [IDE], Small Computer Systems Interface [SCSI], and Serial Advanced Technology Attachment [SATA] disk drives).

Even though SMART attributes are vendor dependent there is a common way of interpreting most SMART attributes. Each SMART attribute has a normalized current value and a threshold value. When the current value exceeds the threshold value, the disk is considered as failed. The ACNS software monitors the SMART attributes and reports any impending failure through syslog messages, SNMP traps, and alarms.

In ACNS 5.3.1 software and later releases, the output from the `show tech-support` EXEC command also includes SMART information.

## System Logging with Standalone Content Engines

Use the system logging feature to set specific parameters for the system log file (syslog). This file contains authentication entries, 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`.

By default, system logging is enabled on a standalone Content Engine. Table 21-6 lists the default settings for system logging.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority of message for console</td>
<td>warning</td>
</tr>
<tr>
<td>Priority of message for log file</td>
<td>debug</td>
</tr>
<tr>
<td>Log file</td>
<td><code>/local1/var/log/syslog.txt</code></td>
</tr>
<tr>
<td>Log file recycle size</td>
<td>0,000,000 bytes</td>
</tr>
</tbody>
</table>

You can use the Content Engine GUI or CLI to configure standalone Content Engines to send varying levels of event messages to disk, console, or remote syslog hosts. See the “Configuring System Logging on Standalone Content Engines” section on page 21-22 for information about how to change the default syslog settings.

Proxy-mode native FTP support is supported in the ACNS 5.3.1 software and later releases. Syslog messages for proxy-mode native FTP support is available in the ACNS 5.3.1 software and later releases. The following is an example of one of these syslog message for proxy-mode native FTP support:

```
CE-FTP_PROXY-3-252009:   Failed to configure FTP Proxy-mode listener on port '[port]'.

Explanation:            Could not start proxy-mode listener for FTP control
                        connection for the specified port. The port is temporarily
                        in an un-bindable state, or is in use by some other
                        application.

Action:                 Check whether the port has been configured for use by a
                        different application. If not, retry the ftp-native
                        incoming proxy command after 2 minutes. If this error
                        repeats frequently, contact Cisco TAC.
```
In the ACNS 5.1.x software and earlier releases, a disk failure syslog message is generated every time that a failed sector is accessed. In the ACNS 5.2.1 software release, support for filtering multiple syslog messages for a single failed sector on an IDE disk was added. In the ACNS 5.3.1 software and later releases, you can filter multiple syslog messages for a single failed section for SCSI disks and SATA disks.

In the ACNS 5.3.1 software and later releases, you can display a list of failed sectors on the Content Engine disks by entering the `show disk failed-sectors` EXEC command:

```
ContentEngine# show disk failed-sectors
List of failed sectors on disk00
--------------------------------
89923
9232112
List of failed sectors on disk01
--------------------------------
<None>
```

To display a list of failed sectors for a only a specific disk drive, specify the name of the disk when entering the `show disk failed-sectors` command. The following example shows how to display a list of failed sectors for disk01:

```
ContentEngine# show disk failed-sectors disk01
```

If there are disk failures, a message is printed notifying you about this situation when you log in to a Content Engine that is running the ACNS 5.3.1 software and later releases.

### Displaying the Current Configuration for System Logging

To display the current syslog host configuration on a standalone Content Engine, enter the `show logging` EXEC command.

```
ContentEngine# show logging
Syslog to host is enabled.
Priority for host logging to 1.2.1.1:514 is set to: warning
Syslog to console is disabled
Priority for console logging is set to: warning
Syslog to disk is enabled
Priority for disk logging is set to: notice
Filename for disk logging is set to: /local1/syslog.txt
Syslog facility is set to syslog
Syslog disk file recycle size is set to 10000000
```

### Configuring System Logging on Standalone Content Engines

You can use the Content Engine GUI or the CLI to configure system logging on standalone Content Engines. As part of the configuration, you specify whether the Content Engine is to send varying levels of messages to any of the following: to disk, to console, or to up to four remote syslog hosts.

From the Content Engine GUI, choose `System > Syslog`. Use the displayed Syslog window to configure system logging for the Content Engine. For more information about how to use the Syslog window, click the Help button in the window to access the context-sensitive help.

From the Content Engine CLI, set specific parameters for the syslog by using the `logging` global configuration commands:

```
ContentEngine(config)# logging ?
```
### Configuring System Logging to the Console

System logging can be configured to send various levels of messages (priority levels) to the console. To configure system logging to the console and to specify the various levels of messages that should be sent to the console, use the `logging console priority` global configuration command:

```
logging console { enable | priority loglevel }
```

Table 21-7 describes these command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>console</td>
<td>Sets system logging to the console.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables system logging to the console.</td>
</tr>
<tr>
<td>priority</td>
<td>Sets which priority level messages to log to the console.</td>
</tr>
<tr>
<td>loglevel</td>
<td>Use one of the following keywords:</td>
</tr>
<tr>
<td></td>
<td>• alert : Immediate action needed. Priority 1.</td>
</tr>
<tr>
<td></td>
<td>• critical: Immediate action needed. Priority 2.</td>
</tr>
<tr>
<td></td>
<td>• debug : Debugging messages. Priority 7.</td>
</tr>
<tr>
<td></td>
<td>• emergency: System is unusable. Priority 0.</td>
</tr>
<tr>
<td></td>
<td>• error : Error conditions. Priority 3.</td>
</tr>
<tr>
<td></td>
<td>• information: Informational messages. Priority 6.</td>
</tr>
<tr>
<td></td>
<td>• notice : Normal but significant conditions. Priority 5.</td>
</tr>
<tr>
<td></td>
<td>• warning : Warning conditions. Priority 4.</td>
</tr>
</tbody>
</table>

Note

Syslog messages from the Content Engine to a remote host are sourced from port 10000 rather than from port 514.

This example shows the last few lines of the syslog.txt file using the `type-tail` EXEC 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.
```
System Logging with Standalone Content Engines

Chapter 21 — Monitoring Standalone Content Engines and Transactions

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

Configuring System Logging to Disk

System logging can be configured to send various levels of messages (priority levels) to disk. To configure system logging to disk and to specify the various levels of messages that should be sent to disk, use the `logging disk priority` global configuration command:

`logging disk {enable \| filename \| priority loglevel \| recycle \| size}`

Table 21-8 describes these command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk</td>
<td>Sets system logging to a disk file.</td>
</tr>
<tr>
<td>enable</td>
<td>Enables system logging to a disk file.</td>
</tr>
<tr>
<td>filename</td>
<td>Sets the name of the syslog file.</td>
</tr>
<tr>
<td>priority</td>
<td>Sets which priority level messages to send to syslog file.</td>
</tr>
<tr>
<td>loglevel</td>
<td>Use one of the following keywords:</td>
</tr>
<tr>
<td></td>
<td>• alert</td>
</tr>
<tr>
<td></td>
<td>• critical</td>
</tr>
<tr>
<td></td>
<td>• debug</td>
</tr>
<tr>
<td></td>
<td>• emergency</td>
</tr>
<tr>
<td></td>
<td>• error</td>
</tr>
<tr>
<td></td>
<td>• information</td>
</tr>
<tr>
<td></td>
<td>• notice</td>
</tr>
<tr>
<td></td>
<td>• warning</td>
</tr>
<tr>
<td>recycle</td>
<td>Overwrites syslog.txt (log file) when the file surpasses the recycle size.</td>
</tr>
<tr>
<td>size</td>
<td>Size of syslog file in bytes (1000000–50000000).</td>
</tr>
</tbody>
</table>

Configuring System Logging to Remote Syslog Hosts

In the ACNS 5.1 software, logging to only a single remote syslog host was supported, and the following two commands were used to configure a single remote syslog host for a standalone Content Engine:

```
ContentEngine(config)# logging host hostname
ContentEngine(config)# logging priority priority
```
In the ACNS 5.2.1 software release and later releases, you can configure a Content Engine to send varying levels of messages to up to four remote syslog hosts. To accommodate this change, the ACNS 5.1.x software logging host priority priority global configuration command is deprecated, and the logging host hostname global configuration command is extended as follows:

```
ContentEngine(config)# [no] logging host hostname [priority priority-code | port port |rate-limit limit]
```

where:

- **hostname** is the hostname or IP address of the remote syslog host.
  Specify up to four remote syslog hosts. To specify more than one syslog host, use multiple command lines; specify one host per command (In the ACNS 5.1.x software and earlier releases, you could only configure a Content Engine to send messages to a single remote syslog host.)
- **priority-code** is the severity level of the message that should be sent to the specified remote syslog host.
  The default priority-code is warning (level 4). Each syslog host can receive a different level of event messages. The different priority codes are shown here:

```
ContentEngine(config)# logging host 1.2.3.4 priority ?
    alert        (1) Immediate action needed
    critical     (2) Critical conditions
    debug        (7) Debugging messages
    emergency    (0) System is unusable
    error        (3) Error conditions
    information  (6) Informational messages
    notice       (5) Normal but significant conditions
    warning      (4) Warning conditions
```

- **port** is the destination port of the remote syslog host to which the Content Engine is to send the messages.
  The default port is port 514. (In releases prior to the ACNS 5.2.1 software release, you could not change the default port. Syslog messages were only sent to port 514 on the specified syslog host)
- **rate-limit** specifies the number of messages that are allowed to be sent to the remote syslog host per second.
  To limit bandwidth and other resource consumption, messages to the remote syslog host can be rate limited. If this limit is exceeded, the specified remote syslog host drops the messages. There is no default rate limit, and by default all syslog messages are sent to all of the configured syslog hosts. If the rate limit is exceeded, a “message of the day” (motd) will be printed for any CLI EXEC shell login.

To configure the Content Engine to send varying levels of syslog messages to up to four external syslog hosts, use the logging host global configuration command. In the following example, the Content Engine is configured to send messages that have a priority code of error (level 3) to the remote syslog host that has an IP address of 172.31.2.160:

```
ContentEngine(config)# logging host 172.31.2.160 priority error
```
System Logging with Standalone Content Engines

Chapter 21 Monitoring Standalone Content Engines and Transactions

Mapping Syslog Priority Levels to RealProxy Error Codes

RealProxy generates error messages and writes them to the RealProxy log file. These error messages are captured by the ACNS software and passed to the system log file. The correspondence between the RealProxy error codes and the syslog priority levels are shown in Table 21-9.

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

Table 21-9 Mapping of RealProxy Error Level to Syslog Priority Level

Using CiscoWorks2000

CiscoWorks2000 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:

- CiscoWorks2000 can list the Content Engine under “Generic SNMP” devices.
- The CiscoWorks2000 inventory module lists the Content Engine with the device name, system name, description (including the software version), uptime, and network interface information.
- The CiscoWorks2000 syslog module can understand Content Engine syslogs.
- The CiscoWorks2000 availability module can track the Content Engine.

You can enable or disable syslog message generation in CiscoWorks2000-compliant format through either the Content Engine GUI or the CLI. For example, to use the Content Engine CLI to configure a CiscoWorks2000 as a remote syslog host, use the `logging host hostname` global configuration command, as described in the “Configuring System Logging on Standalone Content Engines” section on page 21-22.
Monitoring Transactions with Standalone Content Engines

Typically, Content Engine administrators are interested in what types of requests have been made of the Content Engine and what the results of these requests were. For example, if streaming media is a source of revenue for a company, then the company needs a way to track which customer is accessing which content, how long a user viewed the content, and at what viewing quality. Because these companies charge their customers to stream on-demand content and live broadcasts, they must rely on logged information as the basis for billing their customers for their content access services.

The software logs that record requests that are serviced by a Content Engine are referred to as transaction logs. Typical fields in the transaction log are the date and time when a client 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 address.

Transaction logs are generally used for the following purposes:
- Problem identification and solving
- Load monitoring
- Billing
- Statistical analysis
- Security problems
- Cost analysis and provisioning

In the ACNS 5.2.1 software and later releases, Windows Media Services 9 logging is supported. The Windows Media Services 9 Series provides a more robust logging model than Windows Media Services Version 4.1.

You can log to a predefined format (for example, Squid, Extended Squid, or Apache, or a custom transaction log formats that allows you to log additional fields). The contents of the transaction logs can be exported to an external server using FTP at periodic intervals. You can also configure log rotation policies.

**Note**
The term transaction refers to completed successful or failed request for a web resource by a client.
For RTSP, when you choose the Repeat option from the Play menu in the Windows Media player to play media files continuously in a loop, an extra entry is logged in the transaction logs for each playback of the file. This phenomenon occurs mostly with the WMT RTSPU protocol due to the behavior of the player.

### Displaying Statistics for Particular Protocols

For each content transport protocol, there is a corresponding `show protocol-name statistics` EXEC command that displays the statistics for that particular protocol. For example, you can use the `show statistics http` EXEC commands to display important statistical information about the HTTP requests that the Content Engine has serviced.

ContentEngine# show statistics http ?
cluster      Display healing mode statistics
ims          Display If-Modified-Since statistics
miss-reason  Display miss/revalidate/no-store reason statistics
monitor      Display http monitor statistics
object       Display object statistics
performance  Display performance statistics
proxy        Display proxy mode statistics
requests     Display request statistics
savings      Display savings statistics
usage        Display usage statistics

Table 21-10 lists the `show protocol-name statistics` EXEC commands. For detailed information about this commands, see the Cisco ACNS Software Command Reference, Release 5.5 publication.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`show statistics https [error</td>
<td>requests]`</td>
</tr>
<tr>
<td>`show statistics http {cluster</td>
<td>ims</td>
</tr>
<tr>
<td><code>show statistics ftp-native</code></td>
<td>Displays Content Engine FTP native statistics. Includes statistics for FTP native requests (for example, FTP native misses for GET requests).</td>
</tr>
<tr>
<td>`show statistics rtsp {proxy media-real</td>
<td>requests</td>
</tr>
<tr>
<td>`show statistics wmt {all</td>
<td>bytes</td>
</tr>
</tbody>
</table>

**Note**

In the ACNS 5.3.1 software release, the `show statistics ftp` EXEC command was replaced with the `show statistics ftp-over-http` and `show statistics ftp-native` EXEC commands. In the ACNS 5.3.1 software release, the `clear statistics ftp` EXEC command was replaced with the `clear statistics ftp-over-http` and `clear statistics ftp-native` EXEC commands.
The following shows how to display HTTP monitoring statistics for a specific monitored URL (for example, http://www.abccorp.com):

```
ContentEngine# show statistics http monitor
HTTP Monitor URL statistics
---------------------------------------------
Monitor URL = http://www.abccorp.com
Total requests = 2547
Failed requests = 3
Requests above acceptable delay = 1
Minimum response time = 0.072 seconds
Maximum response time = 120.281 seconds
```

The following shows how to display monitoring statistics about the HTTPS requests that a Content Engine has serviced:

```
ContentEngine# show statistics https requests
HTTPS Statistics
Total % of Total
-----------------------------------------------
Total connections: 1328-
Tunneled (CONNECT): 0 0.0
Tunneled (wccp): 0 0.0
SSL terminated: 0 0.0
Connection errors: 0 0.0
Total bytes: 8013157-
Bytes received from client: 1602824 20.0
Bytes sent to client: 6410333 80.0
Bytes received from server: 0 0.0
```

The following shows how to display monitoring statistics for successful and failed outgoing proxy requests:

```
ContentEngine# show statistics http proxy outgoing
HTTP Outgoing Proxy Statistics
IP PORT ATTEMPTS FAILURES
-----------------------------------------------
10.10.10.10 1 49026 49026
```

The following shows how to display statistics about the HTTP requests that a Content Engine has received:

```
ContentEngine# show statistics http requests
Statistics - Requests
Total % of Requests
-----------------------------------------------
Total Received Requests: 525979748 -
Forced Reloads: 501468 0.1
Client Errors: 81834 0.0
Server Errors: 149808 0.0
URL Blocked (Reset): 514998075 97.9
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
If-Range Hits: 32 0.0
HTTP 0.9 Requests: 677 0.0
HTTP 1.0 Requests: 524097101 99.6
HTTP 1.1 Requests: 1881966 0.4
```
HTTP Unknown Requests: 4 0.0
Non HTTP Requests: 0 0.0
Non HTTP Responses: 1380 0.0
Chunked HTTP Responses: 1631953 0.3
Http Miss Due To DNS: 31050 0.0
Http Deletes Due To DNS: 12914 0.0
Objects cached for min ttl: 575986 0.1

The following is sample output from the `show statistics http performance` EXEC command. The command output displays performance statistics regarding the HTTP requests that Content Engine has serviced.

```
ContentEngine# show statistics http performance
Statistics - Performance
Avg Min Max Last
-------------------------------
Requests / Second: - - 677 3
Bytes / Second: - - 5995814 81801
Seconds / Request: 0.067 0.000 15453.547 1.499
Seconds / Hit: 0.308 0.000 979.442 0.158
Seconds / Miss: 0.066 0.000 15453.547 1.572
-------------------------------
Object Size: 150.2 Avg
0 Min
718732317 Max
21386.0 Last
```

The following is sample output of the `show statistics http savings` EXEC command. The command output displays statistics regarding the savings because the Content Engine serviced HTTP requests from its local HTTP cache (cache hits) instead of retrieving the content from the origin web server.

```
ContentEngine# show statistics http savings
Statistics - Savings
Requests Bytes
Total: 525980242 79047534484
Hits: 1966223 19865155481
Miss: 524014019 59182379003
Savings: 0.4 % 25.1 %
```

The following is sample output of the `show statistics rtsp proxy media-real requests` EXEC command. The command output displays RealMedia caching statistics for the RTSP requests that Content Engine has serviced.

```
ContentEngine# show statistics rtsp proxy media-real requests
Media Cache Statistics - Requests
Total % of Requests
-------------------------------
Total Received Requests: 0 -
Demand Cache Hit: 0 0.0
Demand Cache Miss: 0 0.0
Demand Pass-Through: 0 0.0
Live Split: 0 0.0
Live Pass-Through: 0 0.0
```
The following is sample output of the `show statistics rtsp proxy media-real savings` EXEC command. The command output displays the number of media cache hits and misses for RealMedia content, and the amount of savings because content was served from the local cache rather than being retrieved multiple times from the origin streaming server.

```
ContentEngine# show statistics rtsp proxy media-real savings

Media Cache Statistics - Savings

Requests                              Bytes
-----------------------------------------------------------
Total:               525980242                    79047534484
Hits:                 1966223                    19865155481
Miss:               524014019                    59182379003
Savings:                     0.4 %                         25.1 %
```

Using ACNS Software Transaction Logs

Administrators of standalone Content Engines (caching and streaming engines) are often interested in what types of requests have been made of the Content Engine and what the results of these requests were. For example, if streaming media is a source of revenue for a company, then the company needs a way to track which customer is accessing which content, how long a user viewed the content, and at what viewing quality. Because these companies charge their customers to stream on-demand content and live broadcasts, they must rely on logged information as the basis for billing their customers for their content access services.

Standalone Content Engines that are running the ACNS 5.x software can record all errors and access activities. In the ACNS 5.x software releases, each content service module (for example, the HTTP module, the WMT server, the FTP proxy process, and the TFTP server) on the Content Engine provides logs of the requests that were serviced. These logs are referred to as transaction logs.

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 address. Transaction logs are generally used for the following purposes:

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

The reliable production, storage, and management of transaction logs is important in the billing and cost analysis and provisioning cases.

The translog module on the Content Engine handles transaction logging, and supports the following four main logging formats:

- Apache Common Log Format (CLF)
- Squid
- Extended Squid
- World Wide Web Consortium (W3C) Customizable Logging Format
The Apache CLF and Squid formats are fixed formats that correspond to the original applications from which they were derived. The Content Engine supports most of the formats (listed in Table 21-12) defined in the W3C Customizable Logging Format.

The Windows Media Services 9 Series provides a more robust logging model than Windows Media Services Version 4.1. In the ACNS 5.2.1 software and later releases, Windows Media Services 9 logging is supported.

You can log to a predefined format (for example, Squid, Extended Squid, or Apache, or a custom transaction log formats that allows you to log additional fields). The contents of the transaction logs can be exported to an external server using FTP. (In the ACNS 5.3.1 software and later releases, you can also use SFTP to export the contents of the transaction logs to an external server.) You can also configure log rotation policies.

Note: Only one format type can be active at a time. When transaction logging is enabled through the Content Engine GUI, the Squid log format is used.

In the ACNS 5.2.1 software and later releases, there is a transaction log feature that provides a real-time transaction log capability to another device. You can configure a Content Engine to send HTTP transaction log messages to a remote syslog server so that you can monitor HTTP transaction authentication failures in real time. For more information on this topic, see the “Monitoring HTTP Request Authentication Failures in Real Time” section on page 21-48.

By default, transaction logging is disabled on the Content Engine that is running the ACNS software. You can use either the Content Engine GUI or the CLI to enable transaction logging on the Content Engine.

When transaction logging is enabled through the Content Engine GUI, the Squid log format is used as shown in this sample output:

```
ContentEngine(config)# transaction-logs ?
archive Configure archive parameters
enable Enable transaction log feature
export Configure file export parameters
file-marker Add entries to translog indicating the file begin and end
format log file format (default squid)
log-windows-domain Log Windows domain with authenticated username if available
sanitize Mask end user identities in log file
```

Table 21-11 lists the default settings for transaction logging on a standalone Content Engine.

<table>
<thead>
<tr>
<th>Option</th>
<th>Default Setting</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive</td>
<td>Disabled</td>
<td>See the “Archiving the Working Log” section on page 21-46.</td>
</tr>
<tr>
<td>Transaction logging</td>
<td>Disabled</td>
<td>See the “Enabling Transaction Logging” section on page 21-33.</td>
</tr>
<tr>
<td>Export compression</td>
<td>Disabled</td>
<td>See the “Exporting Transaction Log Files” section on page 21-43.</td>
</tr>
<tr>
<td>Export of transaction log</td>
<td>Disabled</td>
<td>See the “Exporting Transaction Log Files” section on page 21-43.</td>
</tr>
<tr>
<td>File marker</td>
<td>Disabled</td>
<td>Use to add entries to the transaction log file to indicate the beginning and the end of a file.</td>
</tr>
<tr>
<td>Sanitized transaction logging</td>
<td>Disabled</td>
<td>See the “Sanitizing Transaction Logs” section on page 21-42.</td>
</tr>
</tbody>
</table>
**Chapter 21 Monitoring Standalone Content Engines and Transactions**

**Monitoring Transactions with Standalone Content Engines**

### Note

SmartFilter provides the information in the transaction log to indicate the categories associated with a URL for an allow transaction as well as a deny transaction. This requires that you use the SmartFilter GUI to enable all SmartFilter logging options (choose the **All** option under **Logging Options** from the SmartFilter GUI).

For more information about working with the ACNS transaction logs, see the following sections:

- **Enabling Transaction Logging, page 21-33**
- **Logging Windows Domain with Authenticated Usernames, page 21-42**
- **Sanitizing Transaction Logs, page 21-42**
- **Exporting Transaction Log Files, page 21-43**
- **Changing the Transaction Logging Export Settings on Standalone Content Engines, page 21-45**
- **Displaying the Transaction Log Configuration for Standalone Content Engines, page 21-51**
- **Restarting Export After Receiving a Permanent Error from the External FTP Server, page 21-45**

#### Enabling Transaction Logging

In the ACNS 5.x software, you can choose the Squid, Extended Squid, or Apache transaction log formats, or you can use a custom log format that allows you to log additional fields. (See **Table 21-12**.)

#### Table 21-11 Default Settings for Transaction Logging for Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Default Setting</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive interval</td>
<td>Every day, every one hour</td>
<td>See the “Archiving the Working Log” section on page 21-46.</td>
</tr>
<tr>
<td>Archive maximum file size</td>
<td>2,000,000 kilobytes</td>
<td>See the “Archiving the Working Log” section on page 21-46.</td>
</tr>
<tr>
<td>Export interval</td>
<td>Every day, every one hour</td>
<td>See the “Exporting Transaction Log Files” section on page 21-43.</td>
</tr>
<tr>
<td>Transaction log format</td>
<td>Squid native log format</td>
<td>See the “Enabling Transaction Logging” section on page 21-33.</td>
</tr>
</tbody>
</table>

#### Table 21-12 List of Supported Transaction Log Formats

<table>
<thead>
<tr>
<th>Style of Transaction Log Format</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squid</td>
<td>See the “Enabling Squid-Style Transaction Logging” section on page 21-34.</td>
</tr>
<tr>
<td>Extended Squid</td>
<td>See the “Enabling Extended Squid-Style Transaction Logging” section on page 21-35.</td>
</tr>
<tr>
<td>Apache</td>
<td>See the “Enabling Apache-Style Transaction Logging” section on page 21-36.</td>
</tr>
<tr>
<td>Custom</td>
<td>See the “Enabling Custom Format Transaction Logging” section on page 21-36.</td>
</tr>
</tbody>
</table>
Note

Only one format type can be active at a time. When transaction logging is enabled through the Content Engine GUI, the Squid log format is used.

In the ACNS 5.2.1 software and later releases, the ability to send HTTP transaction log messages to a remote syslog server is supported. This feature allows you to monitor HTTP transaction authentication failures in real time. The existing transaction logging to the local file system remains unchanged. For more information about real-time transaction logging, see the “Monitoring HTTP Request Authentication Failures in Real Time” section on page 21-48.

Logging FTP Client Usernames

In the ACNS 5.4.1 software and later releases, proxy authentication, that is, request authentication at the Content Engine, was added for nontransparent FTP native requests (nontransparent FTP native requests from such FTP clients as Reflection X clients, WS-FTP clients, and UNIX or DOS command line FTP programs).

If the proxy authentication succeeds for an FTP client, the username that was supplied by the client is logged in the transaction log if the Content Engine has been configured to use one of the following transaction logging formats:

- Extended-Squid logging
- Custom logging

Note

With custom transaction logging, you must include the \%u format-specifier in the transaction-logs format custom global configuration command in order for the supplied username to be logged in the transaction log.

If the proxy authentication fails for an FTP client, the authentication failures are logged in the syslog for monitoring purposes. For more information about enabling and configuring transaction logging on a standalone Content Engine, see the “Enabling Transaction Logging” section on page 21-33.

Enabling Squid-Style Transaction Logging

To enable transaction logging in Squid-style format, enter the transaction-logs format squid global configuration command. 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.

The Squid log file format is as follows:

time elapsed remote host code/status bytes method URL rfc931 peer status/peer host 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 -
```

In the ACNS 5.4.1 software and later releases, FTP proxy authentication is supported. If the FTP proxy authentication succeeds for a particular client (for example, a Reflection X, WS-FTP client, or a UNIX or DOS command line FTP program), the username that is provided by that client during the FTP proxy authentication process is logged in the transaction log if the Extended-squid logging or Custom logging have been configured on the Content Engine.
Squid transaction logs are a valuable source of information about cache workloads and performance. Table 21-13 lists the fields associated with the Squid-style format.

Table 21-13 Squid-Style Format Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>UNIX time stamp as Coordinated Universal Time (UTC) seconds with a millisecond resolution.</td>
</tr>
<tr>
<td>Elapsed</td>
<td>Length of time in milliseconds that the cache was busy with the transaction.</td>
</tr>
<tr>
<td>Note</td>
<td>Entries are logged after the reply has been sent, not during the lifetime of the transaction.</td>
</tr>
<tr>
<td>Remote host</td>
<td>IP address of the requesting instance.</td>
</tr>
<tr>
<td>Code/status</td>
<td>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.</td>
</tr>
<tr>
<td>Bytes</td>
<td>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.</td>
</tr>
<tr>
<td>Method</td>
<td>Request method to obtain an object for example, GET.</td>
</tr>
<tr>
<td>URL</td>
<td>URL requested.</td>
</tr>
<tr>
<td>Rfc931</td>
<td>Contains the authentication server’s identification or lookup names of the requesting client. This field will always be a “-” (dash).</td>
</tr>
<tr>
<td>Peer status/Peer host</td>
<td>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. The hostname may also be numerical.</td>
</tr>
<tr>
<td>Type</td>
<td>Mime-Type of the object as seen in the HTTP reply header. In the ACNS 5.x software, this field will always contain a “-” (dash).</td>
</tr>
</tbody>
</table>

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.

Enabling Extended Squid-Style Transaction Logging

To enable transaction logging in Extended Squid format, enter the `transaction-logs format extended-squid` global configuration command. 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 21-13) is used to log the authorized user. This field always contains a “-” (dash) if no user information is available.

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

1012429341.115 100 172.16.100.152 TCP_MISS/302 184 GET http://www.cisco.com/cgi-bin/login myloginnname DIRECT/www.cisco.com -
Enabling Apache-Style Transaction Logging

To enable transaction logging in Apache style format, enter the transaction-logs format apache global configuration command. The Apache-style log file format is as follows:

remotehost rfc931 authuser date request status bytes

An Apache-style log file format 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
```

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.

Table 21-14 lists the fields associated with the Apache Common Log file format.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remotehost</td>
<td>Remote hostname or IP address.</td>
</tr>
<tr>
<td>Rfc931</td>
<td>Contains the authentication server’s identification or lookup names of the requesting client. This field will always contain a “-” (dash).</td>
</tr>
<tr>
<td>Authuser</td>
<td>Username that the user entered for authentication purposes. This will be a “-” (dash) if no user information is available.</td>
</tr>
<tr>
<td>Date</td>
<td>Date and time of the request.</td>
</tr>
<tr>
<td>Request</td>
<td>First line of the request.</td>
</tr>
<tr>
<td>Status</td>
<td>HTTP status code, for example, 200.</td>
</tr>
<tr>
<td>Bytes</td>
<td>Content length of the document transferred.</td>
</tr>
</tbody>
</table>

Enabling Custom Format Transaction Logging

To log additional fields that are not included in the predefined native Squid or Extended Squid formats, or the Apache Common Log file (CLF) format, use the transaction-logs format custom log-format-string global configuration command.

```
ContentEngine(config)# transaction-logs format custom log-format-string
```

log-format-string is a quoted string of tokens that specifies the custom format. This log format string can contain the tokens listed in Table 21-15, and mimics the Apache log format string.

The log format string can contain these literal characters that are copied into the log file:

- Double backslashes (\) can be used to represent a literal backslash.
- A backslash followed by a single quote (\`) can be used to represent a literal single quote. A literal double quote cannot be represented as part of the log format string.
- The control characters \t and \n can be used to represent a tab and a new line character, respectively.
In the ACNS 5.3.1 software and later releases, invalid custom log format strings cannot be configured. However, software releases prior to Release 5.3 allow you to configure invalid custom log format strings. Therefore, when you upgrade a Content Engine from the ACNS 5.2 software to the ACNS 5.3 software or later releases, any invalid custom log formats that had been configured are deleted.

The following example shows how to specify a custom log format string to generate the well-known Apache Common Log file format:

\texttt{transaction-logs format custom "\{\%{d}t/\%{b}t/\%{Y}t:\%H\t:%M\t:%S\t:%z\t\}r \%s \%b \%{Referer}i \%{User-Agent}i"}

The following example of the transaction log entry in the Apache Common Log file format is configured using the preceding custom format string:

\texttt{[11/Jan/2003:02:12:44 -0800] "GET http://www.cisco.com/swa/i/site_tour_link.gif HTTP/1.1" 200 3436 "http://www.cisco.com/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"}

The custom format currently supports the following request headers:
- User-Agent
- Referer
- Host
- Cookie

The output of each of the following Request, Referer, and User-Agent format tokens specified in the custom log format string is always enclosed in double quotation marks in the transaction log entry:

\texttt{\%r}
\texttt{\%{Referer}i}
\texttt{\%{User-Agent}i}

The \%{Cookie}i format token is generated without the surrounding double quotation marks, because the Cookie value itself can contain double quotes. The Cookie value can contain multiple attribute-value pairs that are separated by spaces. For this reason, it is recommended that when the Cookie format token is used in a custom format string, it be positioned as the last field in the format string. This positioning of the Cookie format token allows it to be more easily parsed by transaction log reporting tools. Alternatively, if you use the format token string “\%{Cookie}i\””, the Cookie header can be surrounded by single quotes.

Table \texttt{21-15} lists the acceptable format tokens for the log format string. The “...” portion of the format tokens shown in Table \texttt{21-15} represents an optional condition. This portion of the format token can be left blank, as in \%a. If an optional condition is included in the format token and the condition is met, then what is shown in the Value column of Table \texttt{21-15} is included in the transaction log output. If an optional condition is included in the format token but the condition is not met, the resulting transaction log output is replaced with a dash (-). The form of the condition is a list of HTTP status codes, which may or may not be preceded by an exclamation point (!).

- The exclamation point is used to negate all of the status codes that follow it, meaning that the value associated with the format token is logged if none of the status codes listed after the ! match the HTTP status code of the request.
- If any of the status codes listed after the ! match the HTTP status code of the request, then a dash (-) is logged.
For example, “%400,501{User-Agent}i” logs the User-Agent header value on 400 errors and 501 errors (Bad Request, Not Implemented) only; “%!200,304,302{Referer}i” logs the Referer header value on all requests that did not return a normal status.

Table 21-15  Custom Log File Format String Values

<table>
<thead>
<tr>
<th>Format Token</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%...a</td>
<td>IP address of the requesting client.</td>
</tr>
<tr>
<td>%...A</td>
<td>IP address of the server from which the object was served (for example, the origin server or the outgoing proxy in the case of a cache miss or 0.0.0.0 in the case of a cache hit).</td>
</tr>
<tr>
<td>%...b</td>
<td>Bytes sent, excluding HTTP headers.</td>
</tr>
<tr>
<td>%...c</td>
<td>Connection status when response is completed, where: X = Connection was aborted before the response was completed. + = Connection can be kept alive after the response is sent. – = Connection is closed after the response is sent.</td>
</tr>
<tr>
<td>%...f</td>
<td>Filename.</td>
</tr>
<tr>
<td>%...h</td>
<td>Remote host (IP address of the requesting client is logged).</td>
</tr>
<tr>
<td>%...H</td>
<td>Request protocol.</td>
</tr>
<tr>
<td>%...{Foobar}i</td>
<td>Contents of Foobar: header lines in the request that is sent to the server. The value of Foobar can be one of the following headers: User-Agent, Referer, Host, or Cookie.</td>
</tr>
<tr>
<td>%...l</td>
<td>Remote log name. Not implemented on the Content Engine, so a dash (-) is logged.</td>
</tr>
<tr>
<td>%...m</td>
<td>Request method.</td>
</tr>
<tr>
<td>%...p</td>
<td>Canonical port of the server servicing the request. Not applicable on the Content Engine, so a dash (-) is logged.</td>
</tr>
<tr>
<td>%...P</td>
<td>Process ID of the child that serviced the request.</td>
</tr>
<tr>
<td>%...q</td>
<td>Query string (that is preceded by a ? if a query string exists; otherwise, it is an empty string).</td>
</tr>
<tr>
<td>%...r</td>
<td>First line of the request.</td>
</tr>
<tr>
<td>%...s</td>
<td>Status. The translog code always returns the HTTP response code for the request.</td>
</tr>
<tr>
<td>%...t</td>
<td>Time in common log time format (or standard English format).</td>
</tr>
<tr>
<td>%...{format}t</td>
<td>Time in the form given by the format token specified in Table 21-16.</td>
</tr>
<tr>
<td>%...T</td>
<td>Time consumed to serve the request in seconds (a floating point number with 3 decimal places).</td>
</tr>
</tbody>
</table>
Table 21-15  Custom Log File Format String Values (continued)

<table>
<thead>
<tr>
<th>Format Token</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%...u</td>
<td>Remote user. In the ACNS 5.4.1 software and later releases, FTP proxy authentication is supported. If the FTP proxy authentication succeeds for a particular client (for example, a Reflection X, WS-FTP client, or a UNIX or DOS command line FTP program), the username that is provided by that client during the FTP proxy authentication process is logged in the transaction log if the Extended-squid logging or Custom logging have been configured on the Content Engine. For the Custom transaction logging format, you must include the %u format-specifier when you configure the <code>transaction-logs format custom</code> command.</td>
</tr>
<tr>
<td>%...U</td>
<td>URL path requested, not including query strings.</td>
</tr>
<tr>
<td>%...v</td>
<td>Value of the host request header field reported if the host appeared in the request. If the host did not appear in the host request header, the IP address of the server specified in the URL is reported.</td>
</tr>
</tbody>
</table>

Table 21-16 specifies the format token for the date and time of the format token %...{format}t that is listed in Table 21-15.

Table 21-16  Format Token for Date and Time

<table>
<thead>
<tr>
<th>Format Token</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%a</td>
<td>Abbreviated weekday name.</td>
</tr>
<tr>
<td>%A</td>
<td>Full weekday name.</td>
</tr>
<tr>
<td>%b</td>
<td>Abbreviated month name.</td>
</tr>
<tr>
<td>%B</td>
<td>Full month name.</td>
</tr>
<tr>
<td>%c</td>
<td>Date and time representation.</td>
</tr>
<tr>
<td>%C</td>
<td>Century number (year/100) as a 2-digit integer.</td>
</tr>
<tr>
<td>%d</td>
<td>Day of the month as a decimal number (01—31).</td>
</tr>
<tr>
<td>%D</td>
<td>Equivalent to %m/%d/%y. (In countries other than the USA, %d/%m/%y is rather common. This means that in international context this format is ambiguous and should not be used.)</td>
</tr>
<tr>
<td>%e</td>
<td>Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space.</td>
</tr>
<tr>
<td>%G</td>
<td>ISO 8601 year with the century as a decimal number. The 4-digit year corresponding to the ISO week number (see %V). This has the same format and value as %y, except that if the ISO week number belongs to the previous or next year, that year is used instead.</td>
</tr>
<tr>
<td>%g</td>
<td>Like %G, but without century; that is, with a 2-digit year (00—99).</td>
</tr>
<tr>
<td>%h</td>
<td>Equivalent to %b.</td>
</tr>
<tr>
<td>%H</td>
<td>Hour as a decimal number using a 24-hour clock (00—23).</td>
</tr>
<tr>
<td>%I</td>
<td>Hour as a decimal number using a 12-hour clock (01—12).</td>
</tr>
<tr>
<td>%j</td>
<td>Day of the year as a decimal number (001—366).</td>
</tr>
</tbody>
</table>
The W3C Customizable Logging Format is limited in that it was defined from the HTTP web server perspective and does not offer certain web cache-specific custom options such as those supplied by the fixed Squid format. Consequently, additional format tokens which are extensions to the W3C
Customized Logging Format, are available in the ACNS 5.3.1 software or later releases to support additional Cisco and Squid-like customized logging fields. These new format tokens provide support for Squid-like logging format from within the W3C customizable token set.

In the ACNS 5.3.1 software and later releases, the following transaction logging support is available:

- Support for the Extended Squid-equivalent tokens that were not supported by the W3C format
- Support for an additional hierarchy token that treats a configured HTTP outgoing proxy (“http outgoing-proxy”) as a Squid-style “DEFAULT_PARENT” hierarchy event

In the ACNS 5.3.1 software release, the W3C Customizable Logging Format was extended to include support for the following special token sequence:

```
%...{<translog-token>}C
```

The “...” is optional. If specified, it can be a sequence of conditional HTTP response codes separated by commas. The “C” is an uppercase C and defines the extended customizable behavior token set, for which tokens are defined by the `<translog-token>` directive, which is a two-character token directive.

Table 21-17 lists the existing and new `<translog-token>` directives from the Extended Squid format, which are not immediately supported by the W3C definitions; they are supported in the ACNS 5.3.1 software and later releases.

<table>
<thead>
<tr>
<th>Format Token</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%...{es}C</td>
<td>Current time presented as the number of seconds that have elapsed since the Epoch (Jan. 1st. 1970).</td>
</tr>
<tr>
<td>%...{em}C</td>
<td>Current number of milliseconds that have elapsed since the Epoch (Jan. 1st. 1970).</td>
</tr>
<tr>
<td>%...{te}C</td>
<td>Number of milliseconds that have elapsed until the request was completed.</td>
</tr>
<tr>
<td>%...{rd}C</td>
<td>Squid-like cache-status code string (for example, TCP_HIT and TCP_CLIENT_REFRESH_MISS).</td>
</tr>
<tr>
<td>%...{cs}C</td>
<td>Number of bytes sent to the client (including the protocol headers).</td>
</tr>
<tr>
<td>%...{rh}C</td>
<td>Strict Squid-style hierarchy as it applies to the Content Engine.</td>
</tr>
<tr>
<td>%...{rH}C</td>
<td>Extended Squid-style hierarchy. Same as “%...{rh}C” except when an outgoing-proxy is explicitly defined and is used to satisfy a request, then the “DEFAULT_PARENT/proxy_ip_address” is logged instead of the “DIRECT/origin_server_ip_address.”</td>
</tr>
<tr>
<td>%...{rt}C</td>
<td>Mime-Type of the object in the response, as specified by any protocol headers which define such. In the ACNS 5.x software, this field will always contain a “-” (dash).</td>
</tr>
<tr>
<td>%...{ru}C</td>
<td>URL being requested, including any additional query strings.</td>
</tr>
<tr>
<td>%...{as}C</td>
<td>Application specific information. Certain request handling applications might want to log a certain string here, which is supported as part of the Squid format specification. For example, SmartFilter URL filtering will log information where this token sequence is used.</td>
</tr>
</tbody>
</table>

In addition to the tokens listed in Table 21-17, you can condense multiple `%...{xx}C` style tokens into a single embedded token sequence within the `%...{xx}C` style. To condense multiple style tokens into a single embedded token sequence, you must specify multiple tokens within the `{ }` braces and prefix each token with the ‘%’ symbol. For example:
Monitoring Transactions with Standalone Content Engines

%{rh}C %{rt}C %{as}C

can be re-expressed in a condensed embedded token format as the following:

%{rh %rt %as}C

The command line syntax will accept single tokens represented as:

%{rh}C

and

%{rh}C

as equivalents.

Any character that is not part of an embedded token sequence (for example, the space character) is repeated verbatim in the output file.

The following is an example of Extended Squid defined within the W3C Customizable Logging format specification:

%{es}C.%{em}C %{te}C %a %{rd} C/%s %{cs}C %m %{ru}C %u %{rh}C %{rt}C %{as}C

The following is an example of an Extended Squid-like format that specifies that user-readable time-stamps are used instead of Squid’s seconds-since-epoch time-stamp format, and that a configured out-going proxy (as specified by “%...{RH}C”) is logged:

[%{d/%b/%Y:%H:%M:%S %z}t] %{te}C %a %{rd} C/%s %{cs}C %m %{ru}C %u %{RH}C %{rt}C %{as}C

Unknown or unsupported translog tokens are not logged within the log file. All characters outside of a token specification sequence are repeated verbatim within the log file.

Logging Windows Domain with Authenticated Usernames

If your Content Engine is configured for NTLM authentication and uses the Extended Squid-style or custom format, the transaction-logs log-windows-domain global configuration command records the Windows domain name and username in the username field of the transaction log. If the domain name is available, both the domain name and the username are recorded in the username field, in the form domain\username. If only the username is available, only the username is recorded in the username field. If neither a domain name nor a username is available, a dash (-) is recorded in the field.

The Windows domain name that is used for NTLM authentication appears in the username field of the transaction log. The username appears in the format domain\username in the Extended Squid-style and custom transaction log formats that contain the username using the %u format token. (The %u format token specifies the day of the week as a decimal; the range is 1 to 7 with Monday being 1.)

To negate logging NTLM parameters to the transaction log in Extended Squid-style or Custom formats, enter the no transaction-logs log-windows-domain global configuration command.

Sanitizing Transaction Logs

You can disguise the IP address and usernames of clients in the transaction log file. The default is that transaction logs are 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.

You can enable the sanitize feature in transaction logging, through the following interfaces:
• Content Engine GUI—From the Content Engine GUI, choose **Cache > Transaction Logs**. Check the **Transaction Log Enable** check box to activate transaction logging on the Content Engine, and then click the **Sanitize transaction logs** radio button to enable the sanitize feature. Click **Update** to apply the settings.

• Content Engine CLI —Use the **transaction-logs sanitized** global configuration command.

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

The **no** form of this command disables the sanitize feature. The **transaction-logs sanitize** command does not affect the client IP (\%a) value associated with a custom log format string that is configured with the CLI, that is, configured with the **transaction-logs format custom string** global configuration command in which **string** is the quoted log format string that contains the custom log format. To hide the identity of the client IP in the custom log format, either hard code 0.0.0.0 in the custom log format string or exclude the \%a token, which represents the client IP, from the format string.

**Exporting Transaction Log Files**

To facilitate the postprocessing of cache log files, you can export transaction logs to an external host. This feature allows 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. In the ACNS 5.3.1 software and later releases, you can also use SFTP to export the contents of the transaction logs to an external host.

The log files automatically have this filename:

```
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 the 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 or STFP Servers**

You can use the Content Engine GUI or CLI to export transaction logs to external FTP or SFTP servers:

From the Content Engine GUI, choose **Caching > Transaction Logs**. Use the displayed Transaction Logs window to export transaction logs to an FTP or SFTP server. For more information about how to use the Transaction Logs window, click the **HELP** button in the window.

To use the Content Engine CLI to enable exporting of transaction logs to an external FTP or SFTP server, follow these steps:

**Step 1** To enable exporting of transaction logs to external FTP or SFTP servers, use the **transaction-logs export enable** global configuration command.

**Step 2** Specify the following information for each target FTP server:
Chapter 21  Monitoring Standalone Content Engines and Transactions

Monitoring Transactions with Standalone Content Engines

ContentEngine(config)# transaction-logs export ftp-server {hostname | server-ip-address}
login password directory

where:

- **hostname or server-ip-address** is the hostname or IP address for the FTP server.
  The Content Engine translates the hostname with a DNS lookup and then stores the IP address in the configuration.
- **login** is the the user login to the target FTP server.
- **password** is the user password to target FTP server.
- **directory** is the target directory path on the specified FTP server to which the exported files (transaction files) are to be written. Specify the name of a working directory that will contain the transaction logs. Use a fully qualified path or a relative path for the user login.

**Note** The user that you specified in the **login** option of this command must have write permission to the specified directory.

In this example, the Content Engine is configured to export its transaction logs to two FTP servers:

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

**Step 3** Export transaction logs to an external SFTP server.

ContentEngine(config)# transaction-logs export sftp-server {hostname | server-ip-address}
login password directory

where:

- **hostname or server-ip-address** is the hostname or IP address for the SFTP server.
  The Content Engine translates the hostname with a DNS lookup and then stores the IP address in the configuration.
- **login** is the the user login to the target SFTP server (less than 40 characters).
- **password** is the user password to target SFTP server (less than 40 characters)
- **directory** is the target directory path on the specified SFTP server to which the exported files (transaction files) are to be written. Specify the name of a working directory that will contain the transaction logs. Use a fully qualified path or a relative path for the user login.

**Step 4** Compress the archived log files into gzip format before exporting them to external FTP or SFTP servers.

ContentEngine(config)# transaction-logs export compress

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 and SFTP export servers also require less bandwidth during export.
Changing the Transaction Logging Export Settings on Standalone Content Engines

After you have specified the transaction logging export configuration (as described in the “Enabling Transaction Logging”), you can change a username, password, or directory, as follows:

- Change the current configuration for an FTP server by using the `transaction-logs export ftp-server` global configuration command.
- Change the current configuration for an SFTP server by using the `transaction-logs exportsftp-server` global configuration command.

As the following example shows, you can change the username, password, or directory by reentering the entire line using the new parameters (for example, mynewname, mynewpass, or newftpdirectory):

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

To delete an FTP server from the current configuration:

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

To delete an SFTP server from the current configuration:

```
ContentEngine(config)# no transaction-logs export sftp-server sftphostname
```

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

When an FTP server returns a permanent error to a standalone 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` EXEC command displays the current state of readiness for transaction log export.

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` global configuration command. The `show statistics transaction-logs` EXEC 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 Put Failures:0
  Retry Attempts:0
  Retry Successes: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
```

## Archiving the Working Log

Depending upon where the sysfs is mounted, the following log files are logged to a working log on the local disk of a standalone Content Engine as follows:

- WMT logs are logged to a working log on the local disk in one of these files:
  - `/local1/logs/export/working.log`
  - `/local2/logs/export/working.log`

- RealProxy logs are logged to a working log on the local disk in one of these files:
  - `/local1/logs/real-proxy/working.log`
  - `/local2/logs/real-proxy/working.log`

You can specify the interval at which the working log should be cleared by moving the data to an archive log. The archive log files are located on the local disk in the `/local1/logs/` or `/local2/logs/` directory depending upon where the sysfs is mounted.

Because multiple archive files are saved, the filename includes the time stamp when the file was archived. Because the files can be exported to an FTP or SFTP server, the filename also contains the IP address of this Content Engine.

The archive filename format is:

```
celog_IPADDRESS_YYYYMMDD_HHMMSS.txt
```

To archive the working logs, use the `transaction-logs archive` global configuration commands.

```
transaction-logs archive interval seconds
transaction-logs archive interval every-day {at hour:minute | every hours}
transaction-logs archive interval every-hour {at minute | every minutes}
transaction-logs archive interval every-week [on weekdays at hour:minute]
```

Table 21-18 describes these command parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>archive</td>
<td>Configures archive parameters.</td>
</tr>
<tr>
<td>interval</td>
<td>Determines how frequently the archive file is to be saved.</td>
</tr>
<tr>
<td>seconds</td>
<td>Frequency of archiving in seconds (120–604800).</td>
</tr>
<tr>
<td>every-day</td>
<td>Archives using intervals of 1 day or less.</td>
</tr>
<tr>
<td>at</td>
<td>Specifies the local time at which to archive each day.</td>
</tr>
<tr>
<td>hour:minute</td>
<td>Time of day at which to archive in local time (hh:mm).</td>
</tr>
<tr>
<td>every</td>
<td>Interval in hours. Interval aligns with midnight.</td>
</tr>
</tbody>
</table>
Chapter 21      Monitoring Standalone Content Engines and Transactions

Monitoring Transactions with Standalone Content Engines

Table 21-18   Parameters of the transaction-logs archive CLI Command (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hours</em></td>
<td>Number of hours for daily file archive.</td>
</tr>
<tr>
<td>1</td>
<td>Hourly</td>
</tr>
<tr>
<td>12</td>
<td>Every 12 hours</td>
</tr>
<tr>
<td>2</td>
<td>Every 2 hours</td>
</tr>
<tr>
<td>24</td>
<td>Every 24 hours</td>
</tr>
<tr>
<td>3</td>
<td>Every 3 hours</td>
</tr>
<tr>
<td>4</td>
<td>Every 4 hours</td>
</tr>
<tr>
<td>6</td>
<td>Every 6 hours</td>
</tr>
<tr>
<td>8</td>
<td>Every 8 hours</td>
</tr>
<tr>
<td><em>every-hour</em></td>
<td>Archives using intervals of 1 hour or less.</td>
</tr>
<tr>
<td><em>at</em></td>
<td>Time to archive at each hour.</td>
</tr>
<tr>
<td><em>minute</em></td>
<td>Minute alignment for the hourly archive (0–59).</td>
</tr>
<tr>
<td><em>every</em></td>
<td>Interval in minutes for hourly archive that aligns with the top of the hour.</td>
</tr>
<tr>
<td><em>minutes</em></td>
<td>Number of minutes for hourly archive.</td>
</tr>
<tr>
<td>10</td>
<td>Every 10 minutes</td>
</tr>
<tr>
<td>15</td>
<td>Every 15 minutes</td>
</tr>
<tr>
<td>2</td>
<td>Every 2 minutes</td>
</tr>
<tr>
<td>20</td>
<td>Every 20 minutes</td>
</tr>
<tr>
<td>30</td>
<td>Every 30 minutes</td>
</tr>
<tr>
<td>5</td>
<td>Every 5 minutes</td>
</tr>
<tr>
<td><em>every-week</em></td>
<td>Archives using intervals of 1 or more times a week.</td>
</tr>
<tr>
<td><em>on</em></td>
<td>(Optional) Day of the week on which to archive.</td>
</tr>
<tr>
<td><em>weekdays</em></td>
<td>Weekdays on which to archive. One or more weekdays can be specified.</td>
</tr>
<tr>
<td>Fri</td>
<td>Every Friday</td>
</tr>
<tr>
<td>Mon</td>
<td>Every Monday</td>
</tr>
<tr>
<td>Sat</td>
<td>Every Saturday</td>
</tr>
<tr>
<td>Sun</td>
<td>Every Sunday</td>
</tr>
<tr>
<td>Thu</td>
<td>Every Thursday</td>
</tr>
<tr>
<td>Tue</td>
<td>Every Tuesday</td>
</tr>
<tr>
<td>Wed</td>
<td>Every Wednesday</td>
</tr>
<tr>
<td><em>at</em></td>
<td>(Optional) Local time at which to archive each day.</td>
</tr>
<tr>
<td><em>hour:minute</em></td>
<td>Time of day at which to archive in local time (hh:mm).</td>
</tr>
<tr>
<td><em>max-file-size</em></td>
<td>Sets maximum archive file size.</td>
</tr>
<tr>
<td><em>filesize</em></td>
<td>Maximum archive file size in kilobytes (1000–2000000).</td>
</tr>
</tbody>
</table>

Disabling Transaction Logging Export on Standalone Content Engines

To disable the transaction logging export feature on a standalone Content Engine while retaining the transaction logging export configuration (for example, such configuration information about the FTP or SFTP servers as their IP addresses), use the `no` form of the `transaction-logs export enable` global configuration command:

ContentEngine(config)# no transaction-logs export enable
Monitoring HTTP Request Authentication Failures in Real Time

In the ACNS 5.2.1 software and later releases, the ability to send HTTP transaction log messages to a remote syslog server is supported. This feature allows you to monitor the remote syslog server for HTTP request authentication failures in real time. This real-time transaction log feature allows you to monitor transaction logs in real time for particular errors such as HTTP request authentication errors. The existing transaction logging to the local file system remains unchanged.

**Note**
Syslog is UDP so message transport to remote syslog host is not reliable.

To support the real-time transaction logging feature, the following CLI commands are supported in the ACNS 5.2.1 software and later releases:

```
[no] transaction-logs logging enable
[no] transaction-logs logging host [hostname | ipaddr] [port port-num rate-limit msgs-per-sec]
[no] transaction-logs logging facility fac-name
[no] transaction-logs logging entry-type entry-type [request-auth-failures | all]
```

These CLI commands allow you to specify a remote syslog host that will receive transaction log messages in real time. The configurable rate-limiting option (the `rate-limit` option) was added to limit the rate at which the transaction logger is allowed to send messages to the remote syslog server (messages per second). You can configure the Content Engine to send transaction log messages in real time to one remote syslog host.

The following are the default settings for the real-time transaction logging feature:

- The real-time transaction logging feature is disabled (the `no transaction-logs logging enable` global command).
- No remote syslog server is specified (the `no transaction-logs logging host` global configuration command).
- No logging facility is specified (the `no transaction-logs logging facility` global configuration command).
- The default facility is “*” to use the facility associated with the transaction logging module that is the “user” facility (user process).
- The default entry type is request-auth-failures. (For more information, see the “Specifying the Transaction Log Entry Type when Logging to a Remote Syslog Host” section on page 21-50.)
- The defaults for the `transaction-logs logging` option are:
  - Port 514 is used. This port is a well-known port for system logging.
  - The rate-limit is set to 0, which means no rate limit. The range is 1 to 10,000 messages per second.

The message format of the transaction log entry to the remote syslog host is the same as in the transaction log file and prepended with Cisco’s standard syslog header information:

```
Apr 22 20:10:46 ce-host cache:%CE-TRNSLG-6-460012: <translog formatted msg>
```

where:

- `ce-host` is the hostname or IP of the Content Engine that is sending the message.
- `cache` is the name of the process on the Content Engine that is sending the message.
- `%CE-TRNSLG-6-460012` is the Cisco standard formatted syslog header.
• `<translog formatted msg>` is the transaction log message as it appears in the transaction log file.

To include the username and domain name in the transaction log, use the following Content Engine CLI command:

```
ContentEngine(config)# transaction-logs log-windows-domain
```

This will cause the Windows domain name that is used for NTLM authentication to appear in the username field of the transaction log. The username appears in the format domain\username in the Extended Squid-style and custom transaction log formats that contain the username using the `%u` format token. Proxy request authentication failures in the HTTP transaction logs are reported as 401/407 errors and include the username. This type of error indicates an HTTP authentication failure. These errors are also included in the system syslog.

### Configuring the Remote Syslog Host for Real-Time Transaction Logging

To configure a standalone Content Engine to send transaction log messages in real time to a remote syslog host, use the `transaction-logs logging host` global configuration command.

In the ACNS 5.2.1 software and later releases, Content Engine system syslog messages are supported to report communication errors with the remote syslog host that is configured for transaction logging. These syslog messages are in the error message range: `$CE-TRNSLG-6-460013` to `$CE-TRNSLG-3-460016`. The last error message (`$CE-TRNSLG-3-460016`), shows level 3 (for error-level messages) instead of 6 (for information-level messages). Information-level messages are reported when messages are dropped due to rate limiting and the number of dropped messages are reported.

### Configuring a Transaction Log Facility when Logging to a Remote Syslog Host

To configure a transaction log facility when logging transactions to a remote syslog host, use the `transaction-logs logging facility` global configuration command.

```
ContentEngine(config)# transaction-logs logging facility ?
    auth Authorization system
    daemon System daemons
    kern Kernel
    local0 Local use
    local1 Local use
    local2 Local use
    local3 Local use
    local4 Local use
    local5 Local use
    local6 Local use
    local7 Local use
    mail Mail system
    news USENET news
    syslog Syslog itself
    user User process
    uucp UUCP system
```

To create a separate log on the remote syslog host for real-time transaction log entries, configure a unique facility (for example, `local1`) on the Content Engine.

```
ContentEngine(config)# transaction-logs logging facility local1
```
On the remote syslog host, configure messages from this unique facility to be logged in a separate file. The following is an example if the remote syslog host is a UNIX server:

1. Edit the /etc/syslog.conf file to add the following line to direct local1 information-level messages to the /var/log/translog-messages file that is associated with the local1 facility:

   local1.=info /var/log/translog-messages

2. Modify the following line in the /etc/syslog.conf file to exclude these information-level messages from the standard output file (the /var/log/messages file):

   *.info;mail.none;news.none;authpriv.none;cron.none;local1.none /var/log/messages

On a UNIX system, help on the syntax for the /etc/syslog.conf file is under the command man syslog.conf.

On a UNIX system, the port for the syslog daemon is defined in /etc/services:

   syslog      514/udp

If a port other than port 514 is configured on the syslog host, you must configure the same port on the Content Engine (the transaction-logs logging host {hostname | ipaddr} [port port-num] global configuration command).

### Specifying the Transaction Log Entry Type when Logging to a Remote Syslog Host

In the ACNS 5.2.1 software and later releases, you can configure the Content Engine to send only the transactions associated with HTTP request authentication failures, or to send all of the transactions.

Typically, organizations are interested in only HTTP request authentication failures for security purposes. By monitoring these types of authentication failures in real time, it enables organizations to identify which end users failed to be authenticated through the Content Engine.

ContentEngine(config)# transaction-logs logging entry-type ?
   all   Log all transaction messages to remote syslog host
   request-auth-failures Log transactions CE failed to authenticate with the auth server

Only the authentication failure transaction that is associated with an end user who is attempting to contact the authentication server is logged. The pending transactions that are waiting for a response from the transaction that contacted the authentication servers are not logged. This approach provides you with the information needed to determine which user fails to authenticate with the Content Engine and minimizes the traffic to the syslog host. In order for you to track which users failed to authenticate, you must configure a transaction log format that logs the username by configuring either Extended Squid-style format or the custom log format with the custom format token %u. For more information about specifying the format of the transaction log, see the “Enabling Transaction Logging” section on page 21-33.

When the transaction-logs logging enable global configuration command is specified, the logging of only HTTP request authentication failures is the default. If you want to change this default and log all transactions, then you must enter the transaction-log logging entry-type all global configuration command on the Content Engine. However, if you log all transactions, there may be a significant UDP drop rate if your syslog host cannot handle the rate of incoming traffic.
Displaying the Transaction Log Configuration for Standalone Content Engines

To display information about the current configuration of transaction logging on a standalone Content Engine, use the `show transaction-log` or `show transaction-logging` EXEC commands. Both of these EXEC commands display the same output. The transaction log file information is displayed for HTTP and WMT caching proxy transactions, as well as TFTP and ICAP transactions. See sample output below.

For security reasons, passwords are never displayed in the output of the `show transaction-log` EXEC command.

```
ContentEngine# show transaction-log
Transaction log configuration:---------------------------------------
Logging is enabled.
End user identity is visible.
File markers are disabled.
Archive interval: every-day every 1 hour
Maximum size of archive file: 2000000 KB
Log File format is squid.
Windows domain is not logged with the authenticated username

Exporting files to ftp servers is disabled.
File compression is disabled.
Export interval: every-day every 1 hour

HTTP Caching Proxy logging to remote syslog host is disabled.
Remote syslog host is not configured.
Facility is the default "*" which is "user".
Log HTTP request authentication failures with auth server to remote syslog host.

HTTP Caching Proxy Transaction Log File Info
  Working Log file - None existing

WMT MMS Caching Proxy/Server Transaction Log File Info
  Working Log file - size : 584
    age: 404
  Archive Log file - mms_export_10.1.1.21_20040622_230415 size: 584
  Archive Log file - mms_export_1.1.1.1_20040623_205825 size: 584
Translog directory doesn’t exist. Maybe because /local1 has no sysfs mounted.

TFTP Transaction Log File Info
  Working Log file - size : 88
    age: 403
  Archive Log file - tftp_server_10.1.1.21_20040622_230415 size: 88
  Archive Log file - tftp_server_1.1.1.1_20040623_205826 size: 88

ICAP Transaction Log File Info
  Working Log file - size : 61
    age: 403
  Archive Log file - icap_10.1.1.21_20040622_230415 size: 61
  Archive Log file - icap_1.1.1.1_20040623_205826 size: 61
```
Monitoring the Performance of Specific URLs

In the ACNS 5.2.3 software and later releases, the ability to configure a Content Engine to monitor the performance of specific URLs is supported. The Content Engine maintains statistics about the various response characteristics for each of the monitored URLs. (You can use the new `show statistics http monitor` command to view these statistics, as described later in this section.)

Use the `http monitor url url` global configuration command to specify up to 10 URLs that you want a Content Engine to monitor.

```
ContentEngine(config)# http monitor url ?
     WORD  URL for monitoring
```

The `http monitor url url` command has two command options, the `acceptable-delay` and `interval` options. As the following sample output indicates, the `acceptable-delay` option is used to specify the acceptable delay in seconds (the maximum number of seconds that the specified monitored URL should be retrieved within). The default acceptable delay is 60 seconds.

```
ContentEngine(config)# http monitor url http://www.abc.com/ accept-delay ?
    <1-3600>  Acceptable delay in seconds
```

As the following sample command output indicates, the `acceptable-delay` option is used to specify the acceptable delay, which is the maximum number of seconds that the specified URL should be retrieved within.

```
ContentEngine(config)# http monitor url http://www.abc.com/ accept-delay 100
```

```
Note  If you use the `http monitor url url` command to configure the same URL with a different interval or acceptable-delay setting, the most recently configured setting takes precedence and overrides any previously configured settings for that particular URL.
```

As the following sample command output indicates, the `interval` option specifies the monitoring interval (that is, how frequently the Content Engine should monitor requests for a specific URL). The monitoring interval is specified in seconds. The default monitoring interval is 60 seconds.

```
ContentEngine(config)# http monitor url http://www.abc.com/ accept-delay 100 interval ?
    <1-3600>  Monitor interval in seconds
```

In the following example, the Content Engine is configured to monitor the URL named `http://www.abc.com/` using the default values (an interval of 60 seconds and an acceptable delay of 60 seconds):

```
http monitor url http://www.abc.com/
```

In the following example, the Content Engine is configured to monitor the URL named `http://www.abc.com/`. The Content Engine is configured to wait up to 100 seconds for the URL to be retrieved and to monitor requests for this URL every 100 seconds.

```
ContentEngine(config)# http monitor url http://www.abc.com/ accept-delay 100 interval 100
```
If it takes more than 100 seconds for the URL to be retrieved, the specified acceptable delay is exceeded. The Content Engine tracks the response time (minimum and maximum delay time) as well as the number of times that the acceptable delay is exceeded for a particular URL. These statistics are shown in the output from the new `show statistics http monitor` EXEC command.

To display statistics for the monitored URLs, enter the `show statistics http monitor` EXEC command.

```
ContentEngine# show statistics http monitor
HTTP Monitor URL statistics
---------------------------
Monitor URL                              = http://www.abc.com/
Total requests                           = 118
Failed requests                          = 30
Requests above acceptable delay          = 37
Minimum response time                    = 8.183 seconds
Maximum response time                    = 210.021 seconds

Monitor URL                              = http://www.abccorp.com/
Total requests                           = 275
Failed requests                          = 44
Requests above acceptable delay          = 26
Minimum response time                    = 0.071 seconds
Maximum response time                    = 164.061 seconds
```

In this command output the following applies:

- Failed requests are requests that did not succeed (for example, the request failed to resolve the domain name of that URL).
- Requests above acceptable delay are the requests that took longer than the specified acceptable delay (the maximum number of seconds specified by the acceptable-delay setting).

The output of the `show running-configuration` EXEC command includes information about the URL monitoring configuration. In the following excerpt from the `show running-configuration` command output, this particular information is highlighted in bold:

```
ContentEngine# show running-configuration
! ACNS version 5.4
!
!
hostname sust-7320-ce1
!
http persistent-connections timeout 300
http proxy incoming 8080
http proxy outgoing preserve-407
http tcp-keepalive enable
http monitor url http://www.abc.com/ interval 100 acceptable-delay 100
http monitor url http://www.abccorp.com/
!
ftp proxy incoming 8080
!
clock timezone US/Eastern -5 0
!
!
!
!
!
!
```

Only the non-default values are displayed in the output from the `show running-configuration` command. Consequently, because the Content Engine was configured to use the default values to monitor the URL http://www.abccorp.com, the sample output does not display these values for that URL.

To display a list of monitored URLs, including the interval and acceptable delay setting for each monitored URL, enter the `show http monitor` EXEC command.
ContentEngine# show http monitor

Monitor URL: http://www.abc.com/
Monitor Interval: 100
Acceptable Delay: 100

Monitor URL: http://www.abccorp.com/
Monitor Interval: 60
Acceptable Delay: 60
Troubleshooting

This chapter describes how to troubleshoot locally managed deployments (standalone Content Engines). This chapter contains the following sections:

- **Overview, page 22-1**
- **Troubleshooting with HTTP Statistics Tools, page 22-2**
- **Troubleshooting with the Debug Tool, page 22-4**
- **Troubleshooting URL Performance, page 22-5**
- **Troubleshooting with the HTTP CLI Client, page 22-7**
- **Troubleshooting in Real Time, page 22-9**
- **Troubleshooting with the Log Viewing Tool, page 22-9**
- **Troubleshooting with the Telnet Client, page 22-10**
- **Troubleshooting with Ping, page 22-10**
- **Troubleshooting with Traceroute, page 22-10**
- **Troubleshooting with the WMT Diagnostic Tools, page 22-11**

**Note**

For complete syntax and usage information for the CLI commands used in this chapter, see the *Cisco ACNS Software Command Reference, Release 5.5* publication.

For information about troubleshooting a Content Router, Content Distribution Manager, or a Content Engine that is registered with a Content Distribution Manager (as opposed to standalone Content Engines that are not registered with a Content Distribution Manager), see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*.

**Overview**

Troubleshooting is a process of elimination, testing, and validation. It often requires you to use prior knowledge of network topology and business policies, apply protocol knowledge and application behavior, and work with network and application teams to resolve a problem.
Troubleshooting with HTTP Statistics Tools

Use the `show statistics http savings` EXEC command to verify HTTP bandwidth savings, as shown in the following example:

```
CE# show statistics http savings

Statistics - Savings

Requests                          Bytes
-----------------------------------------------------------
Total:              1559097021                  6421538629735
Hits:              1041965581                  1890163375706
Miss:               517131440                  4531375254029
Savings:                    66.8 %                         29.4 %

```

Use the `show statistics http performance` EXEC command to verify HTTP performance as shown in the following example:

```
CE# show statistics http performance

Statistics - Performance

Requests / Second:          -            -             432             199
Bytes / Second:          -            -        18992735          703016
Seconds / Request:      0.760        0.000     4291967.812           0.191
Seconds / Hit:      0.083        0.000     4291367.331           0.030
Seconds / Miss:      2.124        0.000     4291967.812           0.557
________________________________________________________________________
Object Size:                  4118.8 Avg
                             0 Min
                             2147484671 Max
                             3530.7 Last
```

You can use the Web application troubleshooting tools described in this chapter to search out the root causes of many common problems.
Use the `show statistics http miss-reason` EXEC command to show the reasons for a cache miss, as shown in the following example:

```
CE# show statistics http miss-reason
Statistics - No hit reasons
Reason:                          No. of Requests
-----------------------------------------------
not_in_cache:                  269562
  dmbuf_low:                           0
  none_get_method:                    0
  ftp_not_anonymous:                  0
  http_not_anonymous:                 0
  suspicious_url:                     0
   ie_5_ims:                           0
      has_if_match:                    0
      has_invalid_if_range:            0
      has_if_unmodified_since:         0
        has_invalid_range:             0
      has_more_than_supported_range:   0
         hasPragma_no_cache:          1260
      has_authorization:               0
      has_cache_control_no_cache:      0
          is_https:                        0
      invalid_ims:                      0
      cert_check_fail:                  0
      second_validation:               0
      invalid_im_reply:                 0
      imz_200_reply:                     0
      xfs_open_error:                   650
    hasUnknown_lengthTransfer_pending:  0
   object_in_cache_older_than_clients: 0
object_in_cache_expired_cannot_verified: 0
    different_protocol:                0
      other_error:                     620

Statistics - Validate reasons
Reason:                          No. of Requests
-----------------------------------------------
reval_all:                        0
reval_text:                       0
    max_age:                          0
    min_fresh:                        0
    max_stale:                        0
response_say_so:                  0
    object_expired:                   0
reval_no_cache_req:               0
    rule_refresh:                      0

Statistics - No store reasons
Reason:                          No. of Requests
-----------------------------------------------
dmbuf_low:                          0
  none_get_method:                    0
  ftp_not_anonymous:                  0
  http_not_anonymous:                 0
  suspicious_url:                     0
   has_range:                           0
   has_authorization:                  0
   has_cache_control_no_store:         0
     invalid_ims:                      0
     cert_check_fail:                  0
    second_validation:                 0
    invalid_im_reply:                  0
    invalid_imz_reply:                 0
     url_too_long:                      0
```
Increasing the Cache Hit Rate

Use the following global configuration commands for cache tuning and transport tuning to increase the cache hit rate:

- `http cache-cookies`
- `http cache-cookies`
- `http cache-vary-user-agent enable`
- `http cache-chunk-encoded enable`
- `http smart-range enable`
- `http cache-authenticated {all | basic | ntlm}`
- `http fast-response enable` (application specific, not always necessary; disables Nagle’s algorithm)
- `http tcp-keepalive enable` (forces the Content Engine to send KAL probes so connections are not closed)

Troubleshooting with the Debug Tool

Use the `debug http` EXEC command to troubleshoot issues such as the following:

- When you are not sure what header parameters have been set by the applications, use the `debug http header` command.
- If the Content Engine is caching objects when it is expected not to cache them, use the `debug http hit` command.
If the Content Engine is not caching objects when it is expected to cache them, use the `debug http miss` command.

- If you want to verify Content Engine requests or responses for caching parameters, use the `debug http cache` command.

## Troubleshooting URL Performance

To configure a Content Engine to monitor the performance of specific URLs, use the `http monitor url` global configuration command.

This command enables you to specify up to 10 URLs that you want the Content Engine to monitor and allows you to specify the time interval (in seconds) for monitoring the specified URL(s), as well as the acceptable delay (in seconds) before the URL is retrieved. The Content Engine maintains statistics about the various response characteristics for each of the monitored URLs. (You can use the new `show statistics http monitor` command to view these statistics, as described later in this section.)

```
ContentEngine(config)# http monitor url ?
WORD  URL for monitoring
```

The `http monitor url` command has two command options, the `acceptable-delay` and `interval` options. As the following sample output indicates, the `acceptable-delay` option is used to specify the acceptable delay in seconds (the maximum number of seconds that the specified monitored URL should be retrieved within). The default acceptable delay is 60 seconds.

```
ContentEngine(config)# http monitor url http://www.abc.com/ acceptable-delay ?
<1-3600>  Acceptable delay in seconds
```

As the following sample command output indicates, the `acceptable-delay` option is used to specify the acceptable delay, which is the maximum number of seconds that the specified URL should be retrieved within.

```
<cr>
```

As the following sample command output indicates, the `interval` option specifies the monitoring interval (that is, how frequently the Content Engine should monitor requests for a specific URL). The monitoring interval is specified in seconds. The default monitoring interval is 60 seconds.

```
ContentEngine(config)# http monitor url http://www.abc.com/ acceptable-delay 100 interval ?
<1-3600>  Monitor interval in seconds
```

In the following example, the Content Engine is configured to monitor the URL named “http://www.abc.com/” using the default values (an interval of 60 seconds and an acceptable delay of 60 seconds):

```
http monitor url http://www.abc.com/
```
In the following example, the Content Engine is configured to monitor the URL named “http://www.abc.com/.” The Content Engine is configured to wait up to 100 seconds for the URL to be retrieved and to monitor requests for this URL every 100 seconds.

```text
ContentEngine(config)# http monitor url http://www.abc.com/ acceptable-delay 100 interval 100
```

If it takes more than 100 seconds for the URL to be retrieved, the specified acceptable delay is exceeded. The Content Engine tracks the response time (minimum and maximum delay time) as well as the number of times that the acceptable delay is exceeded for a particular URL. These statistics are shown in the output from the new `show statistics http monitor` EXEC command. (An example of the output from the `show statistics http monitor` EXEC command is provided below.)

To display statistics for the monitored URLs, use the `statistics http monitor` EXEC command. The following example shows the statistics that are reported for each of the monitored URLs:

```text
ContentEngine# show statistics http monitor
HTTP Monitor URL statistics
---------------------------
Monitor URL: http://www.abc.com/
Total requests: 118
Failed requests: 30
Requests above acceptable delay: 37
Minimum response time: 8.183 seconds
Maximum response time: 210.021 seconds
```

In the command output shown above, note the following information:

- “Failed requests” are requests that did not succeed (for example, the request failed to resolve the domain name of that URL).
- “Requests above acceptable delay” are the requests that took longer than the specified acceptable delay (the maximum number of seconds specified by the acceptable-delay setting).

The output of the `show running-configuration` EXEC command includes information about the URL monitoring configuration. In the following excerpt from the `show running-configuration` command output, this particular information is highlighted in bold:

```text
ContentEngine# show running-configuration

hostname sust-7320-ce1
http persistent-connections timeout 300
http proxy incoming 8080
http proxy outgoing preserve-407
http tcp-keepalive enable
http monitor url http://www.abc.com/ interval 100 acceptable-delay 100
http monitor url http://www.abccorp.com/
ftp proxy incoming 8080
```

In the command output shown above, note the following information:

- “Failed requests” are requests that did not succeed (for example, the request failed to resolve the domain name of that URL).
- “Requests above acceptable delay” are the requests that took longer than the specified acceptable delay (the maximum number of seconds specified by the acceptable-delay setting).
Only the non-default values are displayed in the output from the `show running-configuration` command. Consequently, because the Content Engine was configured to use the default values to monitor the URL “http://www.abccorp.com,” the above sample output does not display these values for that URL.

To display a list of monitored URLs, including the interval and acceptable delay setting for each monitored URL, use the `show http monitor` EXEC command:

```
ContentEngine# show http monitor
Monitor URL: http://www.abc.com/
Monitor Interval: 100
Acceptable Delay: 100

Monitor URL: http://www.abccorp.com/
Monitor Interval: 60
Acceptable Delay: 60
```

## Troubleshooting with the HTTP CLI Client

In the ACNS 5.2 software and later releases, an HTTP CLI client is supported. This capability allows you to test connectivity and debug caching issues. The `test-url` EXEC command provides debugging capabilities to the end user from the Content Engine CLI. With this command, you can test the URL accessibility and connectivity through the web server and the proxy server. The functionality of the URL over the HTTP, HTTPS, and FTP protocols can be tested. This command tests the functionality of a URL over the HTTP and FTP proxy servers. The following command output is displayed to the standard output (the console):

```
ContentEngine# test-url?
ftp    For request over FTP
http   For request over HTTP
https  For request over HTTPS
```

Table 22-1 describes the `test-url` commands.

<table>
<thead>
<tr>
<th><strong>Table 22-1</strong> test-url CLI Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>test-url Content Engine CLI Commands</strong></td>
</tr>
<tr>
<td><strong>test-url http test-URL</strong></td>
</tr>
<tr>
<td><code>custom-header</code></td>
</tr>
<tr>
<td><code>head-only</code></td>
</tr>
<tr>
<td><code>use-http-proxy</code></td>
</tr>
</tbody>
</table>
When using the **test-url** EXEC command, you can specify the test URL (**test-url** protocol **test-URL** EXEC command) in either of the following formats:

```
http://domain.com/path or http://user:password@domain.com/path
```

In the following example, the test URL (www.cisco.com) is an HTTP request, and the request succeeds:

```
ContentEngine# test-url http http://www.cisco.com
--17:34:50--  http://www.cisco.com/=> `/dev/null'
Len - 21 , Restval - 0 , contlen - 0 , Res - 134728056
Resolving www.cisco.com... done.
Connecting to www.cisco.com[192.168.0.0]:80... connected.
HTTP request sent, awaiting response...
 1 HTTP/1.1 200 OK
 2 Date: Fri, 25 Jun 2006 17:09:06 GMT
 3 Server: Apache/1.0 (Unix)
 4 Content-Type: text/html
 5 Set-Cookie: CP_GUTC=172.16.0.0.133781088183346835; path=/; expires=T
 19-Jun-29 17:09:06 GMT; domain=.cisco.com
 6 Via: 1.1 Application and Content Networking System Software 5.5.1
 7 Connection: Close
(-1 to go)
  [ == ] 0 --.--K/s
  en - 1185   ELen - 0   Keepalive - 0 [ == ] 56,249 1.38M/s
17:34:50 (1.38 MB/s) - `/dev/null' saved [56249]
```

In the following example, the test URL (gmail.google.com) is an HTTPS request that fails because the host is not found:

```
ContentEngine# test-url https https://gmail.google.com head-only
--17:43:13--  https://gmail.google.com/=> `/dev/null'
Len - 25 , Restval - 0 , contlen - 0 , Res - 134728056
Resolving gmail.google.com... failed: Host not found.
```
In the following example, the test URL is an FTP-over-HTTP request using the Content Engine as the FTP proxy for the request:

```
ContentEngine# test-url ftp ftp://1234567:pAB6rB7x@smartfilter.com use-ftp-proxy ?
ContentEngine# test-url ftp ftp://1234567:pAB6rB7x@smartfilter.com use-ftp-proxy
```

**Troubleshooting in Real Time**

To obtain traffic output in real time, use the `type-tail` EXEC command with the `follow` option. This command can be used to follow any working log or syslog.txt files. You can also use the `tcpdump` EXEC command to capture all of the traffic on a Content Engine interface.

The following example shows real-time output from the working log:

```
CE# type-tail working.log follow
```

```
1143265877.603 18 10.82.163.18 TCP_IMS_HIT/304 148 GET http://www.cisco.com/swa/i/flyout_arrow_red.gif - NONE/ - ALLOW "-"
1143265877.683 29 10.82.163.18 TCP_IMS_HIT/304 149 GET http://www.cisco.com/swa/i/spacer_grey.gif - NONE/ - ALLOW "-"
1143265877.683 29 10.82.163.18 TCP_IMS_HIT/304 148 GET http://www.cisco.com/swa/i/flyout_arrow_red.gif - NONE/ - ALLOW "-"
1143265877.690 6 10.82.163.18 TCP_IMS_HIT/304 149 GET http://www.cisco.com/swa/i/spacer_grey.gif - NONE/ - ALLOW "-"
1143265877.792 45 10.82.163.18 TCP_IMS_HIT/304 148 GET http://www.cisco.com/swa/i/flyout_arrow_red.gif - NONE/ - ALLOW "-"
1143265877.793 45 10.82.163.18 TCP_IMS_HIT/304 149 GET http://www.cisco.com/swa/i/spacer_grey.gif - NONE/ - ALLOW "-"
1143265877.798 5 10.82.163.18 TCP_IMS_HIT/304 148 GET http://www.cisco.com/swa/i/flyout_arrow_red.gif - NONE/ - ALLOW "-"
1143265877.901 44 10.82.163.18 TCP_IMS_HIT/304 149 GET http://www.cisco.com/swa/i/spacer_grey.gif - NONE/ - ALLOW "-"
```

**Troubleshooting with the Log Viewing Tool**

In the ACNS 5.2 software and later releases, you can use the `less` EXEC command to specify the filename of the ACNS software log that you want to view:

```
ContentEngine# less ?
WORD file name
```
Troubleshooting with the Telnet Client

In the ACNS 5.2.1 software and later releases, a Telnet CLI client is supported. This Telnet client allows you to specify a destination port. This allows you to test websites by attempting to connect with Telnet to the website from the Content Engine CLI. To support this feature, the `telnet` EXEC command was added.

```
ContentEngine# telnet ?
  Hostname or A.B.C.D  Remote host or IP address
```

Troubleshooting with Ping

To send echo packets for diagnosing basic network connectivity on networks, use the `ping` EXEC command:

```
ping [hostname | ip-address] | PING options
```

In the ACNS 5.2.1 software and later releases, the functionality of the `ping` EXEC command supports all of the standard ping options as command arguments. (For more information on the standard ping options, see the Linux man page.)

```
ContentEngine# ping ?
  LINE  Destination Host or IP Address (or PING options)
```

To use the `ping` EXEC command with the `hostname` argument, be sure that DNS functionality is configured on your Content Engine. To force the timeout of a nonresponsive host, or to eliminate a loop cycle, press `Ctrl-C`.

In the following example, the host with an IP address of 172.19.131.189 is pinged from the Content Engine CLI:

```
ContentEngine# ping 172.19.131.189
PING 172.19.131.189 (172.19.131.189) from 10.1.1.21 : 56(84) bytes of data.
 64 bytes from 172.19.131.189: icmp_seq=0 ttl=249 time=613 usec
 64 bytes from 172.19.131.189: icmp_seq=1 ttl=249 time=485 usec
 64 bytes from 172.19.131.189: icmp_seq=2 ttl=249 time=494 usec
 64 bytes from 172.19.131.189: icmp_seq=3 ttl=249 time=510 usec
 64 bytes from 172.19.131.189: icmp_seq=4 ttl=249 time=493 usec

--- 172.19.131.189 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.485/0.519/0.613/0.047 ms
```

In the ACNS 5.2.1 software and later releases, the functionality of the `ping` command supports all of the standard ping options. (For more information on the standard ping options, see the Linux man page.)

```
ContentEngine# ping ?
  LINE  Destination Host or IP Address (or PING options)
```

Troubleshooting with Traceroute

Traceroute is a widely available utility on most operating systems. Much like ping, it is a valuable tool for determining connectivity in a network. Ping allows the user to find out if there is a connection between two end systems. Traceroute does this as well, but additionally lists the intermediate routers between the two systems. Users can therefore see the routes that packets can take from one system to another.
To find the route to a remote host, when either the hostname or IP address is known, use the `traceroute` EXEC command.

ContentEngine# traceroute yahoo.com
traceroute to 66.218.71.113 (66.218.71.113), 30 hops max, 38 byte packets
***
***
***
10 p3-3.paloalto-cr2.bbnplanet.net (4.0.26.13) 3.219 ms 2.001 ms 2.097 ms
11 p7-1.paloalto-nbr2.bbnplanet.net (4.0.6.77) 3.133 ms 1.949 ms 2.076 ms
12 p4-0.paloalto-nbr1.bbnplanet.net (4.0.5.65) 2.755 ms 2.204 ms 2.037 ms
13 p1-0.paix-bi2.bbnplanet.net (4.0.6.98) 2.928 ms 2.146 ms 2.334 ms
14 p1-0.xpaix17-level3.bbnplanet.net (4.0.1.70) 3.397 ms 3.631 ms 3.081 ms
15 gige10-0.ipcolo4.SanJose1.Level3.net (64.159.2.42) 3.334 ms 2.999 ms 2.388 ms
16 cust-int.level3.net (64.152.69.18) 3.871 ms 3.031 ms *
17 ge-3-3-0.msr1.pao.yahoo.com (192.168.101.42) 3.695 ms ge-2-3-0.msr2.pao.yahoo.com
(192.168.101.46) 6.998 ms *
18 vl16.bas1.scd.yahoo.com (66.218.64.146) 6.282 ms 5.091 ms 5.162 ms
19 w2.rc.scd.yahoo.com (66.218.71.113) 6.028 ms 5.782 ms 5.544 ms

**Troubleshooting with the WMT Diagnostic Tools**

In the ACNS 5.3.1 software and later releases, you can access the following three WMT diagnostic tools through the Content Engine CLI:

- **asfhead**—Examines the headers of a Windows Media file (for example, an .asf, .wmv, or .wma file).
  
  To access the asfhead tool, enter the `show wmt diagnostics header-info stream-file` EXEC command, as follows:

  ContentEngine# show wmt diagnostics header-info stream-file word

  where `word` is a local file.

- **nschead**—Examines the .nsc file headers.
  
  The .nsc file is created when a multicast station is configured. This tool is useful for multicast debugging. To access the nschead tool, enter the `show wmt diagnostics header-info nscl-filename` EXEC command, as follows:

  ContentEngine# show wmt diagnostics header-info nscl-filename .nsc-filename

The following is a sample annotated output from the `show wmt diagnostics header-info stream-file` EXEC command. In this example, this command is used to display the headers of a .wmv file named 256.wmv. Annotations about this sample command output are highlighted in boldface text and enclosed within parentheses.

ContentEngine# show wmt diagnostics header-info stream-file 256.wmv
Description object:
VBR info object (2 streams) (VBR: variable bit rate file)
streamid 1 bitrate 34319 (0000860f) (in this file there are 2 bit rates)
streamid 2 bitrate 195798 (0002fcd6)
Global properties object:
  GUID = f5 dd df 7e 6c 80 85 46 83 3c c1 23 4b 2f 4f 6b (object ID)
  File Size = 8104309 (object size)
  Total Packets = 2816 (total number of data packets)
  Time_mkin = 0x0000000000001c0debc
  Time_minout = 0x000000000a9e1af60
  Send duration = 2817510000 (Specifies the time needed to send the file in 100-nanosecond units)
  Broadcast Flag = 0
Seekable Flag = 1
Minimum Packet Size = 2877  (Specifies the minimum data packet size in bytes)
Max Packet Size = 2877   (Specifies the maximum data packet size in bytes)
Max Bitrate = 230117  (Specifies the maximum instantaneous bit rate in bits per second for the entire file)

Unknown object:
uuid: b5 3 bf 5f 2e a9 cf 11 8e e3 0 c0 c 20 53 65 len: 0x16
00000000: 11 d2 d3 ab ba a9 cf 11 8e e6 00 c0 c 20 53 65
00000010: 06 00 00 00 00 00
Extended Content Description Object (two entries):
WMFSDKVersion (unicode) 7.00.00.1956
WMFSDKNeeded (unicode) 0.0.0.0000
Codec description object (2 streams) (audio and video codec information)
  Type = 0x2
  Description = Windows Media Audio V7
      32 kbps, 32 kHz, stereo
      ID = 61 01
  Type = 0x1
  Description = Windows Media Video V7
      ID = 57 4d 56 31

Index:
  Index #0
  Interval = 10000000
  Max_pkt# = 9
  Entry = 286
  Stream ID = 0

The following is an excerpt of sample output from the show wmt diagnostics header-info nsc-file EXEC command. In this example, this command is used to display the headers of the .nsc file named testmcast.nsc. Annotations about this sample command output are highlighted in boldface text and enclosed within parentheses.

ContentEngine# show wmt diagnostics header-info nsc-file testmcast.nsc
Press Ctrl-C to abort, if no information is shown within 30 secs.

[Address]
Time To Live=0x00000005 (ttl)
IP Address=233.33.33.33 (Multicast address that is configured.)
IP Port=0x000000D10 (Multicast port that is configured.)
Delivery Mode=0x00000002
NSC Format Version=3.0  (version of NSC format)
[Description]
Description=
Auto Archive=0x00000000
Format1 id 4d2
Description1=http://192.168.192.4:8080  (Source of the media stream, for example, the WMT server/encoder)

header information:
output_head() starting
Description object:
clip: WMT Live testing  (title of the stream, configured on the WMT server/encoder)
author: John Doe  (author of the stream, configured on the WMT server/encoder)
VBR info object  (2 streams)
  streamid 1 bitrate 34995  (000088b3)
  streamid 2 bitrate 253711  (0003df0f)
Global properties object:
    GUID = 33 36 59 e6 6a 7d 4e 49 99 5e cf 71 39 b9 61 54
    (guid: unique identifier of this stream)
Troubleshooting with the WMT Diagnostic Tools

File Size = 2810 (size in bytes)
Total Packets = 4294967295
Time mkin = 0x0000000000c4de45
Time mkout = 0x0000000000000000
Send duration = 0
Broadcast Flag = 1
Seekable Flag = 0
Minimum Packet Size = 1444
Max Packet Size = 1444
Max Bitrate = 288706

Unknown object:
uuid: b5 3 bf 5f 2e a9 cf 11 8e e3 0 c0 c 20 53 65 len: 0x707
00000000: 11 d2 d3 ab ba a9 cf 11 8e e6 00 c0 0c 20 53 65
00000010: 06 00 f1 06 00 00 a9 46 43 7c e0 ef fc 4b b2 29
00000020: 39 72 c6 32 43 83 99 a9 69 52 06 5b 5a 58 00 00
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000a0: 69 52 06 5b 5a 6e 00 00 00 00 00 00 00 00 00 00
000000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000e0: 00 63 17 05 00 00 00 00 00 00 00 00 00 00 00 00
000000f0: c6 7f 86 07 49 83 a3 c7 79 21 b7 33 ad 02 00 00
00000100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000110: 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00
00000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140: 00 00 00 49 00 73 00 56 00 42 00 52 00 00 00 00
00000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160: 00 76 00 69 00 63 00 65 00 43 00 4f 00 6e 00 66
00000170: 00 76 00 69 00 63 00 65 00 43 00 4f 00 6e 00 66
00000180: 00 65 00 70 00 70 00 6c 00 61 00 74 00 65 00 00
00000190: 00 4c 00 32 00 32 00 32 00 32 00 32 00 32 00 02
000001a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001c0: 00 76 00 69 00 63 00 65 00 43 00 4f 00 6e 00 66
000001d0: 00 6f 00 6f 00 6f 00 6f 00 6f 00 6f 00 6f 00 6f
000001e0: 00 6f 00 70 00 70 00 6c 00 61 00 74 00 74 00 65
000001f0: 00 4d 00 32 00 32 00 32 00 32 00 32 00 32 00 02
00000200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Extended Content Description Object (3 entries):
WMFSDKVersion (unicode) 9.00.00.2980
WMFSDKNeeded (unicode) 0.0.0.0000
IsVBR (bool) 0

Codec description object (2 streams)
  Type = 0x2
  Description = Windows Media Audio 9
    32 kbps, 32 kHz, stereo (A/V) 1-pass CBR
    ID = 61 01
  Type = 0x1
  Description = Windows Media Video 9
    ID = 57 4d 56 33

Cisco ACNS Software Configuration Guide for Locally Managed Deployments
Multicast object (len 0x18):
00000000: 50 0f 1d 54 4b 5b cf 11 a8 fd 00 80 5f 5c 44 2b
00000010: 04 00 00 00 0a 00 00 00
ContentEngine#
PART 6

Reference Material for Standalone Content Engines
Content Engine GUI Menu Options

This appendix describes the tabs and subtabs (menu options) that are available from the Content Engine GUI. This is an alternative method that you can use to configure and monitor standalone Content Engines.

This appendix includes the following sections:

- Content Engine GUI Tabs, page A-2
- WCCP Tab and Subtabs, page A-2
- Caching Tab and Subtabs, page A-4
- System Tab and Subtabs, page A-6
- Reporting Tab and Subtabs, page A-7

Note For information about how to access the Content Engine GUI, see the “Logging in to the Content Engine GUI” section on page 4-55. The Content Engine GUI has context-sensitive help that can be accessed by clicking the HELP button at the bottom of the Content Engine GUI window.
Content Engine GUI Tabs

Table A-1 describes the four feature tabs and their associated functions.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCCP</td>
<td>Enables WCCP on the Content Engine and configures WCCP-related parameters and services (for example, configures the Content Engine to support the web cache service).</td>
</tr>
<tr>
<td>Caching</td>
<td>Configures cache-related parameters (for example, content preloading) on the Content Engine.</td>
</tr>
<tr>
<td>System</td>
<td>Configures system-related parameters (for example, access lists, DNS, and Websense server parameters) on the Content Engine.</td>
</tr>
<tr>
<td>Reporting</td>
<td>Displays statistics (for example, disk statistics, performance statistics, and WMT streaming statistics) gathered by the Content Engine.</td>
</tr>
<tr>
<td></td>
<td>Displays a hardware profile (model number, CPU, memory, disks, SCSI, and NICs) of the Content Engine, and the version of the ACNS software that is currently running on the Content Engine.</td>
</tr>
</tbody>
</table>

WCCP Tab and Subtabs

Table A-2 lists the ACNS 5.3.1 software or later features that can be configured from the WCCP tab and subtabs of the Content Engine GUI.

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable WCCP</td>
<td>Enables WCCP on the Content Engine. Use to enable WCCP Version 1 or WCCP Version 2 on the Content Engine.</td>
</tr>
<tr>
<td>Clustering</td>
<td>Sets parameters related to WCCP service clusters.</td>
</tr>
<tr>
<td>Custom Web Cache</td>
<td>(For WCCP Version 2 configuration only)</td>
</tr>
<tr>
<td></td>
<td>Configures the WCCP custom web cache service on the Content Engine. When the Content Engine is configured to support this WCCP Version 2 service, it acts as a transparent forward proxy server for HTTP requests that are transparently redirected to it by any of the WCCP-enabled routers on the specified router list. If the requested content is not in its local cache, the Content Engine retrieves it from the origin web server, stores a local copy for future requests, and sends the client the requested content. Subsequent requests for the same content is served from the local cache on the Content Engine.</td>
</tr>
<tr>
<td>Reverse Proxy</td>
<td>(For WCCP Version 2 configuration only)</td>
</tr>
<tr>
<td></td>
<td>Configures the WCCP reverse proxy service on the Content Engine. When the Content Engine is configured to support this WCCP Version 2 service, it acts as a reverse proxy; the Content Engine acts as a proxy on behalf of the origin web server.</td>
</tr>
</tbody>
</table>
Table A-2  
Content Engine GUI WCCP Subtabs (continued)

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTSP</td>
<td>(For WCCP Version 2 configuration only) Configures the WCCP RTSP media cache service on the Content Engine (a transparent forward proxy server). The Content Engine listens for redirected RTSP requests on the standard RTSP port (default port 554). To configure the Content Engine to listen for intercepted RTSP requests on ports other than the default port (port 554), configure a user-defined WCCP service (services 90-97). If the requested content is not in its local cache, the Content Engine retrieves it from the origin streaming server, stores a local copy for future requests, and sends the client the requested content. Subsequent requests for the same RTSP streaming content is served from the local cache on the Content Engine.</td>
</tr>
<tr>
<td>Services</td>
<td>(For WCCP Version 2 configuration only) Configures the Content Engine to cache web traffic using multiple ports, using the user-defined WCCP services (service 90 to 97). To configure these generic WCCP Version 2 services, use this WCCP Services window.</td>
</tr>
<tr>
<td>Web Cache</td>
<td>(WCCP Version 1 or Version 2 configuration) Configures the WCCP web-cache service (service 0) on the Content Engine (transparent forward proxy server). This service permits the Content Engine to accept transparently redirected HTTP requests on a single port (port 80). If the requested content is not in its local cache, the Content Engine retrieves it from the origin web server, stores a local copy for future requests, and sends the client the requested content. Subsequent requests for the same web content is served from the local cache on the Content Engine. Only a single WCCP router is supported with WCCP Version 1; multiple routers (router list) are supported with WCCP Version 2. To enable the Content Engine to listen for WCCP intercepted HTTP traffic on ports other than the default port, configure the custom-web-cache service or a user-defined WCCP service (services 90 to 97).</td>
</tr>
<tr>
<td>WMT-Streaming</td>
<td>(For WCCP Version 2 configuration only) Configures the WCCP WMT media caching service on the Content Engine (transparent forward proxy server). If the requested content is not in its local cache, the Content Engine retrieves it from the origin streaming server, stores a local copy for future requests, and sends the client the requested content. Subsequent requests for the same WMT streaming content is served from the local cache on the Content Engine.</td>
</tr>
</tbody>
</table>
## Caching Tab and Subtabs

Table A-3 lists the ACNS 5.3.1 software or later features that can be configured from the Caching tab and subtab of the Content Engine GUI.

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth. Cache</td>
<td>Configures cache authentication on the Content Engine. For more information, see the “Configuring Authenticated HTTP Cache Settings” section on page 7-12.</td>
</tr>
<tr>
<td>Bypass</td>
<td>(For WCCP Version 2 configuration only) Configures bypass on the Content Engine. For more information, see the “Configuring Bypass Settings on Standalone Content Engines” section on page 15-3.</td>
</tr>
<tr>
<td>Cache on Abort</td>
<td>Configures cache-on-abort features on the Content Engine. Determines the policy for object caching that the Content Engine should use if the web client aborts the download process. If this option is enabled, the Content Engine uses a selective algorithm to determine whether to continue to cache an object if the web client has aborted the download. If disabled, the Content Engine will always continue to download an object to the cache even if a web client has aborted the download.</td>
</tr>
<tr>
<td>Content Preload</td>
<td>Configures content preloading on the Content Engine. For more information, see the “Configuring Content Preloading for Standalone Content Engines” section on page 11-2.</td>
</tr>
<tr>
<td>Customized Error Page</td>
<td>Creates HTTP customized error pages. If you create these customized pages, then the Content Engine displays the appropriate customized error page instead of the default error message when proxy errors occur. For more information, see the “Creating Custom Message Pages for Standalone Content Engines” section on page 5-14.</td>
</tr>
<tr>
<td>FTP Freshness</td>
<td>Views or configures FTP object freshness factors for FTP-over-HTTP caching for the Content Engine. For more information, see the “Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines” section on page 7-38.</td>
</tr>
<tr>
<td>FTP Proxy</td>
<td>Configures FTP incoming and outgoing proxies for FTP-over-HTTP requests. Proxy mode 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. DNS must be configured in order to support incoming FTP proxy requests. For more information, see the “Configuring Nontransparent FTP-over-HTTP Caching on Standalone Content Engines” section on page 7-38.</td>
</tr>
<tr>
<td>HTTP Freshness</td>
<td>Views or configures freshness factors HTTP objects that are currently cached on the Content Engine. For more information, see the “Configuring HTTP Cache Freshness Settings” section on page 7-9.</td>
</tr>
<tr>
<td>HTTP Proxy</td>
<td>Configures HTTP incoming and outgoing proxies for the Content Engine. Proxy mode enables the Content Engine to operate in environments where Cisco’s WCCP is not enabled, or where client browsers have previously been configured to use a legacy proxy server. DNS must be configured in order to support incoming HTTP proxy requests. For more information, see the “Configuring Nontransparent HTTP Forward Proxy Caching on Standalone Content Engines” section on page 7-8.</td>
</tr>
<tr>
<td>HTTPS Proxy</td>
<td>Configures HTTPS incoming and outgoing proxies. HTTPS proxy mode enables the Content Engine to service HTTPS requests sent by web clients that are configured to use an HTTPS proxy server. DNS must be configured in order to support incoming HTTPS proxy requests. For more information, see the “Configuring HTTPS Proxy Caching for Standalone Content Engines” section on page 7-25.</td>
</tr>
</tbody>
</table>
### Table A-3  Content Engine GUI Caching Subtabs (continued)

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ICP Client</strong></td>
<td>Configures a Content Engine cluster to generate ICP queries before retrieving the requested objects from the Internet. For example, specify how long the Content Engine should wait before retrieving the needed data directly from the Internet. By default, the Content Engine waits for 2 seconds; however, you can change this default. The range is 1 to 30 seconds. For more information, see the “Configuring Standalone Content Engines as ICP Clients” section on page 7-73.</td>
</tr>
<tr>
<td><strong>ICP Server</strong></td>
<td>Configures a Content Engine to act as an ICP server. Specify whether the Content Engine is the parent server or sibling server for the designated ICP client. If the Content Engine is the parent ICP server and cannot satisfy the ICP client’s request, it forwards the request to another server on the Internet. If the Content Engine is a sibling ICP server and cannot satisfy the ICP client’s request, it will send a failed response back to the ICP client. For more information, see the “Configuring Standalone Content Engines as ICP Servers” section on page 7-74.</td>
</tr>
<tr>
<td><strong>LDAP</strong></td>
<td>Configures the Content Engine to use an LDAP server for authentication purposes. To enable the Content Engine to use a specific LDAP server, enter the IP address of the LDAP server, and the port number that the LDAP server will be listening on. The default LDAP port number is 389. LDAP authentication is not performed if no LDAP servers are configured. For more information, see the “Configuring the LDAP Authentication Service” section on page 10-21.</td>
</tr>
<tr>
<td><strong>Native FTP Proxy</strong></td>
<td>Configures FTP incoming and outgoing proxies for FTP native requests from such FTP clients as Reflection X or WS-FTP clients. For more information, see the “Configuring Nontransparent FTP Native Caching” section on page 7-42.</td>
</tr>
<tr>
<td><strong>NTLM</strong></td>
<td>Configures the Content Engine to use an NTLM server for authentication purposes. To enable the Content Engine to use a specific NTLM server, enter the host information. For example, specify the domain name (the domain name in which the user should be authenticated) and the domain server (the IP address or hostname of the NTLM server). NTLM authentication will not be performed if no NTLM servers are configured. For more information, see the “Configuring the NTLM Authentication Service” section on page 10-38.</td>
</tr>
<tr>
<td><strong>Persist. Connect.</strong></td>
<td>Configures persistent connections on the Content Engine. Persistent connections can be set for client-only, server-only, and all connections on the Content Engine. If the Strip NTLM Authentication headers option is turned off in the Content Engine, the NTLM authentication headers will be sent to the client. The Content Engine will not support NTLM authentication if persistent connections for all is turned off. For more information, see the “Configuring Standalone Content Engines to Send out TCP Keepalives” section on page 7-67.</td>
</tr>
<tr>
<td><strong>Proxy Protocols</strong></td>
<td>Configures proxy protocols for the Content Engine. The default behavior is that the Content Engine will retrieve the objects from the origin web server itself, or, if configured to use an outgoing proxy for this protocol, will forward the request to the specified outgoing proxy.</td>
</tr>
<tr>
<td><strong>RADIUS</strong></td>
<td>Configures the Content Engine to use a remote RADIUS server for authentication purposes. Configure the network parameters required to access the remote RADIUS database.</td>
</tr>
<tr>
<td><strong>RealProxy</strong></td>
<td>Enables the RealProxy GUI Administration page. Real Proxy must be configured properly using the in order to start the Real Proxy. For more information, see the “Configuring RealProxy with the RealSystem Administrator GUI” section on page 8-21.</td>
</tr>
<tr>
<td><strong>Transaction Logs</strong></td>
<td>Enables and configures transaction logging. By default, transaction logging is disabled. For more information, see the “Monitoring Transactions with Standalone Content Engines” section on page 21-27.</td>
</tr>
</tbody>
</table>
Table A-3  Content Engine GUI Caching Subtabs (continued)

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Filtering</td>
<td>Configures URL filtering. For more information, see “Configuring URL Filtering on Standalone Content Engines” section on page 11-11.</td>
</tr>
<tr>
<td>WMT-Streaming</td>
<td>Enables WMT and configures WMT streaming parameters. For more information, see the Chapter 9, “Configuring WMT Streaming Media Services on Standalone Content Engines.”</td>
</tr>
</tbody>
</table>

System Tab and Subtabs

Table A-4 lists the ACNS 5.3.1 software or later features that can be configured from the System tab and subtabs of the Content Engine GUI.

Table A-4  Content Engine GUI System Subtabs

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Lists</td>
<td>Manages group name-based access lists on the Content Engine. In environments in which the Content Engine has been deployed as a standalone caching engine, you can use group-based access lists to control which groups of users can view specific content that is served by the Content Engine. For more information, see the “Configuring the RADIUS Authentication Service” section on page 10-19.</td>
</tr>
<tr>
<td>Accounting</td>
<td>Configures authentication, authorization, and accounting (AAA). AAA accounting is the action of keeping track of administrative user activities for system accounting purposes. For more information, see Chapter 18, “Configuring AAA Accounting on Standalone Content Engines.”</td>
</tr>
<tr>
<td>Authentication</td>
<td>Configures how the Content Engine is to authenticate and authorize administrative users. For more information, see Chapter 17, “Configuring Administrative Login Authentication and Authorization on Standalone Content Engines.”</td>
</tr>
<tr>
<td>Basic Networking</td>
<td>Adjusts the network settings (the default gateway and the hostname) for the Content Engine.</td>
</tr>
<tr>
<td>CDP</td>
<td>Configures the Cisco Discovery Protocol (CDP) for the Content Engine. By default, CDP is enabled on a Content Engine.</td>
</tr>
<tr>
<td>DNS</td>
<td>Configures DNS name servers for the Content Engine to use for domain name resolution, and enable DNS caching on the Content Engine. For more information about this topic, see the “Configuring DNS Caching for Standalone Content Engines” section on page 7-62.</td>
</tr>
<tr>
<td>File System</td>
<td>Displays information about the currently configured file systems.</td>
</tr>
<tr>
<td>NTP</td>
<td>Sets the Content Engine time and date.</td>
</tr>
<tr>
<td>Real Subscriber</td>
<td>Displays Real Subscriber activities. Real Subscriber is only supported with Content Engines that are registered with a Content Distribution Manager. Real Subscriber is not supported with standalone Content Engines.</td>
</tr>
<tr>
<td>Routing</td>
<td>Adds a new entry to the routing table.</td>
</tr>
<tr>
<td>Rules Template</td>
<td>Configures or modifies the Rules Template on the Content Engine. For more information, see Chapter 13, “Configuring the Rules Template on Standalone Content Engines.”</td>
</tr>
<tr>
<td>SNMP</td>
<td>Configures SNMP on the Content Engine. For more information, see the “Monitoring Standalone Content Engines with SNMP” section on page 21-2.</td>
</tr>
</tbody>
</table>
### Table A-5 Content Engine GUI Reporting Subtabs

<table>
<thead>
<tr>
<th>Subtab Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Stats</td>
<td>Displays general CFS disk statistics for the Content Engine.</td>
</tr>
<tr>
<td>Hardware Info</td>
<td>Displays detailed information about the Content Engine hardware components (for example, model number and RAM size).</td>
</tr>
<tr>
<td>IMS Stats</td>
<td>Displays If-Modified-Since (IMS) request activity (for example, the total number of requests from clients to the Content Engine).</td>
</tr>
<tr>
<td>Java Monitor</td>
<td>Displays the Content Engine resources; displays a graphical depiction of Content Engine utilization.</td>
</tr>
<tr>
<td>Performance</td>
<td>Displays performance statistics for the Content Engine.</td>
</tr>
<tr>
<td>Requests</td>
<td>Displays statistics on miscellaneous HTTP request data (for example, the number of forced reloads).</td>
</tr>
<tr>
<td>Savings</td>
<td>Displays the number of requests that have been served by the Content Engine.</td>
</tr>
<tr>
<td>TCP</td>
<td>Displays the amount of requests that have been served by the Content Engine.</td>
</tr>
<tr>
<td>Usage</td>
<td>Displays the resource utilization statistics for the Content Engine.</td>
</tr>
<tr>
<td>WMT-Streaming</td>
<td>Displays WMT streaming statistics.</td>
</tr>
</tbody>
</table>
Reference Material for Standalone Content Engine Deployments

This appendix contains important reference material that is pertinent to configuring and monitoring standalone Content Engines.

This appendix includes the following sections:

- Supported Network Protocols, page B-2
- Supported Streaming Media Protocols, page B-3
- Supported WCCP Services, page B-3
- Matrix of Supported Caching, Filtering, and Authentication Methods, page B-7
- Supported Access Control and Filtering Services for Content Requests, page B-7
- ACNS Software CLI Command Modes for Standalone Content Engines, page B-8
- ACNS Software CLI Online Help and Keyboard Shortcuts, page B-10
- Unusable Multicast Address Assignments, page B-11

---

**Note**

The term *standalone Content Engines* is used throughout this guide to refer to Content Engines that the ACNS administrators have intentionally not registered with a Content Distribution Manager (if there is one in the network) so that they can configure, manage, and monitor these Content Engines as standalone devices. This information is the focus of this guide. For information about configuring, managing, or monitoring Content Engines that are registered with a Content Distribution Manager, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5.*
Supported Network Protocols

Table B-1 lists the network protocols that standalone Content Engines, which are running the ACNS 5.1 software and later releases, can use to serve content to the web client. Support for HTTP, FTP, TFTP, HTTPS and the IETF standard RTP/RTSP protocols is included as part of the ACNS software product. Support for the following two product features each need a separate license:

- The WMT product feature that offers Windows Media Services requires a WMT license.
- The RealNetworks’ RealProxy feature that uses RealNetworks’ RTSP protocol, which includes proprietary extensions to the standard IETF standard RTSP protocol, requires a RealProxy license.

To enable the licensed WMT product feature on a Content Engine, you must have a WMT license key, which is supplied on a certificate shipped with the Content Engine. If you are downloading the ACNS 5.x software, you can purchase a WMT license through the Cisco.com website. For more information, see the “Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines” section on page 9-14.

To enable the licensed RealProxy product feature on a Content Engine, you must have a RealProxy license key, which is supplied on a certificate shipped with the Content Engine. If you are downloading the ACNS 5.x software, you can purchase a RealProxy license through the Cisco.com website. For more information, see the “Configuring RealMedia Services” section on page 8-9.

Table B-1  Supported Network Protocols

<table>
<thead>
<tr>
<th>Network Protocol</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>The main protocol used on the web for communication between web browsers and web servers. There are two commonly implemented HTTP versions today: HTTP 1.0 and HTTP 1.1. The ACNS 5.x software supports both HTTP 1.0 and HTTP 1.1. See the “Configuring HTTP Caching for Standalone Content Engines” section on page 7-7.</td>
</tr>
<tr>
<td>FTP</td>
<td>The prevalent file transfer protocol before HTTP became the main protocol. FTP is typically used in applications such as software distribution applications. For more background information, see the “Overview of FTP Caching with Standalone Content Engines” section on page 7-36. For information about configuring FTP caching on a standalone Content Engine, see the “Configuring FTP Caching for Standalone Content Engines” section on page 7-36.</td>
</tr>
<tr>
<td>TFTP</td>
<td>A simple file transfer protocol that is similar to FTP but has fewer features and is less complicated. This protocol is still being used by devices that need to download boot images or configurations using a simple protocol (for example, it is used by routers and IP phones). For more background information, see the “Configuring the TFTP Server and Gateway for Standalone Content Engines” section on page 7-58.</td>
</tr>
</tbody>
</table>
Table B-2 lists the streaming media protocols, control channels, the corresponding data format, and transport types that can be used to deliver streaming media files with standalone Content Engines.

Table B-2  Streaming Media Protocols

<table>
<thead>
<tr>
<th>Streaming Media Protocol</th>
<th>Control Channel</th>
<th>Data Format</th>
<th>Transport Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Media format</td>
<td>TCP</td>
<td>RTSP</td>
<td>UDP, TCP, HTTP, IP multicast</td>
</tr>
<tr>
<td>RealNetworks media format</td>
<td>TCP</td>
<td>RTSP, PNA</td>
<td>UDP, TCP, HTTP, IP multicast</td>
</tr>
</tbody>
</table>

1. RTSP=Real Time Streaming Protocol
2. User Datagram protocol
3. PNA=Progressive Networks Audio

Supported WCCP Services

The type of WCCP services supported by a standalone Content Engine and a WCCP-enabled router varies based on whether WCCP Version 1 or Version 2 is used, as indicated in Table B-3. All of the services except for the standard web-cache service (service 0) requires that WCCP Version 2 (as opposed to WCCP Version 1) is running on the router and the standalone Content Engine for a particular WCCP service to be supported. These services are called predefined WCCP services.
Some of the WCCP services that the routers can supply have a well known set of criteria and have a predefined service identifier. For instance, the standard web-cache service (service 0) is a currently supported web-caching service that has a predefined service identifier and well know set of criteria (for example, redirects HTTP requests to port 80). See Table B-3 for a list of WCCP services that have a predefined service identifier (service number).

Other services, that are not well known, may be defined by the configuration of a set of criteria and assigned a service identifier to it. This configuration can be performed by the Content Engines (web caches) that form a particular WCCP service group so that they can define what the criteria is as new WCCP services are developed, but is gated by the routers that need to enable the acceptance of the particular service identifier that the Content Engines are defining the criteria for. WCCP services 90 to 97 are provided to allow you to configure such user-defined services, define the criteria for the new service, and then assign it a service number (identifier).

Note In the ACNS software releases earlier than the ACNS 5.2 software release, a maximum of eight active WCCP services were supported by a WCCP Version 2-enabled router and a Content Engine. In the ACNS 5.2.1 software release, up to 25 active WCCP Version 2 services are supported. In the ACNS 5.2 software release, there are 17 WCCP Version 2 services that can be configured. In the ACNS 5.3.1 software release, there are 18 WCCP Version 2 services that can be configured. In the ACNS 5.3.1 software release, service 83 was added.

<table>
<thead>
<tr>
<th>Service Number</th>
<th>Service Name</th>
<th>Type of Service</th>
<th>Service Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>web-cache</td>
<td>Predefined</td>
<td>Web-caching service that permits WCCP Version 1 or Version 2-enabled router to redirect HTTP traffic to a single port on the Content Engine. The Content Engine is functioning as a transparent forward proxy server. Only a single WCCP-enabled router is supported with WCCP Version 1. Multiple WCCP-enabled routers (those on the router list) are supported with WCCP Version 2. The Content Engine listens for redirected HTTP requests on the standard HTTP port (default port 80). To enable the Content Engine to listen for WCCP-intercepted HTTP traffic on ports other than the default port, configure the custom-web-cache service or a user-defined WCCP service (services 90 to 97). See the “Configuring the Standard Web-Cache Service (Service 0) on a Router” section on page 6-27.</td>
</tr>
<tr>
<td>53</td>
<td>dns</td>
<td>Predefined</td>
<td>DNS-caching service that permits WCCP Version 2-enabled routers to redirect client requests transparently to a Content Engine for the Content Engine to resolve the DNS name. After the Content Engine resolves the DNS name, it stores the resolved DNS name locally so that it can use the resolved names for future DNS requests. See the “Configuring the DNS Caching Service (Service 53) on a Router” section on page 6-28.</td>
</tr>
<tr>
<td>60</td>
<td>ftp-native</td>
<td>Predefined</td>
<td>Caching service that permits WCCP Version 2-enabled routers to redirect FTP native requests transparently to a single port on the Content Engine. The Content Engine retrieves the requested FTP content, stores a copy locally, and serves the requested content to the requester. See the “Configuring the FTP-Native Caching Service (Service 60) on a Router” section on page 6-29. Note In the ACNS 5.3.1 software release, the name of this WCCP service was changed from ftp to ftp-native to clearly differentiate between FTP native requests and FTP-over-HTTP requests. Service 60 (the ftp-native service) only applies to transparent redirection of FTP native requests and does not apply to FTP-over-HTTP requests.</td>
</tr>
</tbody>
</table>
### Table B-3 Supported WCCP Services with Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Service Number</th>
<th>Service Name</th>
<th>Type of Service</th>
<th>Service Description</th>
</tr>
</thead>
</table>
| 70             | https-cache  | Predefined      | Caching service that permits WCCP Version 2-enabled routers to intercept port 443 TCP traffic and redirect this HTTPS traffic to the Content Engine (acting as a transparent forward proxy server that is configured for HTTPS transparent caching). The Content Engine retrieves the requested content, stores a copy locally (HTTPS transparent caching), and serves the requested content to the client.

In the ACNS 5.2.1 software release, another interception mode (the accept-all mode) was added for the WCCP https-cache service. This mode was added to support the filtering of HTTPS traffic. This mode works the same way as the traditional WCCP services (for example, the web-cache service that intercepts all web traffic by default).

By default, the Content Engine accepts all HTTPS traffic.

```
ContentEngine(config)# wccp https-cache ?
accept-all       Accept all https traffic by default
mask             Specify mask used for CE assignment
router-list-num  Router list number
```

If the `wccp https-cache accept-all` global configuration command is used, the HTTPS cache (the Content Engine that has the https-cache service configured and enabled) operates in accept-all mode (all HTTPS traffic is intercepted by the Content Engine); otherwise, the Content Engine (the HTTPS cache) works in accept-only mode, as in the ACNS 5.1.x software.

The Content Engine listens for redirected HTTPS requests on the standard HTTPS port (default port 443). To intercept HTTPS traffic on ports other than the default port, configure a user-defined WCCP service (services 90 to 97). See the “Configuring the HTTPS-Cache Service (Service 70) on a Router” section on page 6-29.

| 80             | rtsp         | Predefined      | Media-caching service that permits WCCP Version 2-enabled routers to redirect RTSP client requests transparently to a single port on a Content Engine (RealMedia transparent caching).

The Content Engine listens for redirected RTSP requests on the standard RTSP port (default port 554). To intercept RTSP traffic on ports other than the default port (port 554), configure a user-defined WCCP service (services 90 to 97). To configure transparent interception of RTSP requests from RealMedia clients, you only need to configure the rtsp service (service 80) on the WCCP Version 2-enabled router. For information about configuring service 80 on a router, see the “Configuring the RTSP Service (Service 80) on a Router” section on page 6-30.

In contrast, you must configure the rtsp service (service 80) as well as the wmt-rtspu service (service 83) on the WCCP Version 2-enabled router to configure transparent interception of WMT RTSP requests. For information about configuring service 83 on a router, see the “Configuring the WMT-RTSPU Service (Service 83) on a Router” section on page 6-31.
### Supported WCCP Services with Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>Service Number</th>
<th>Service Name</th>
<th>Type of Service</th>
<th>Service Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>wmt-rtspu</td>
<td>Predefined</td>
<td>Media-caching service that permits WCCP Version 2-enabled routers to redirect RTSP requests from Windows Media Services 9 clients (Windows Media 9 players) transparently to a single port on a Content Engine (port 5005). The Content Engine listens for redirected RTSP requests from Windows Media 9 Players on the standard Windows Media RTSPU port (default port 5005). To intercept RTSP requests from Windows Media 9 players on ports other than the default port, configure a user-defined WCCP service (services 90 to 97). To configure transparent interception of WMT RTSP requests, you must configure the rtsp service (service 80) as well as the wmt-rtspu service (service 83) on the WCCP Version 2-enabled router. For information about configuring service 83 on a router, see the “Configuring the WMT-RTSPU Service (Service 83) on a Router” section on page 6-31.</td>
</tr>
<tr>
<td>90–97</td>
<td>User-configurable</td>
<td>User-defined</td>
<td>Eight user-defined (dynamic) WCCP services that each support multiple ports (up to eight ports per WCCP service). In order to configure these services (services 90 to 97), you must create one port list for each user-defined service that will be used (for example, create port list number 1 for service 90). The port list contains the port numbers on which the WCCP Version 2-enabled router will support WCCP redirection for that particular WCCP service. When configuring these user-defined services, you must specify whether the traffic is to be redirected to the HTTP caching application, HTTPS caching application, or the streaming application on the Content Engine. To configure the Content Engine to cache web traffic using multiple ports, configure a user-defined WCCP service (services 90 to 97) Use these user-defined WCCP services to support WCCP redirection of HTTP, HTTPS, and RTSP requests on multiple ports (up to eight ports per service) for standard WCCP services (for example, the https-cache, rtsp, mmst, and reverse-proxy services) that ordinarily only support a single port. See the “Configuring User-Defined WCCP Services (Services 90–97) on a Router” section on page 6-31.</td>
</tr>
<tr>
<td>98</td>
<td>custom-web-cache</td>
<td>Predefined</td>
<td>Caching service that permits WCCP Version 2-enabled routers to redirect HTTP traffic to a Content Engine on multiple ports other than port 80. The Content Engine is functioning as a transparent forward proxy server. This service allows you to support WCCP redirection of HTTP requests on multiple ports (up to eight ports) without having to configure a user-defined WCCP service (services 90 to 97). See the “Configuring the Custom-Web-Cache Service (Service 98) on a Router” section on page 6-32.</td>
</tr>
<tr>
<td>99</td>
<td>reverse-proxy</td>
<td>Predefined</td>
<td>Caching service that permits WCCP Version 2-enabled routers to redirect HTTP reverse proxy traffic to a Content Engine (a transparent reverse proxy server) on a single port (port 80). To intercept reverse proxy traffic on ports other than the default port (port 80), configure a user-defined WCCP service (services 90 to 97). See the “Configuring the Reverse-Proxy Service (Service 99) on a Router” section on page 6-33.</td>
</tr>
</tbody>
</table>
Matrix of Supported Caching, Filtering, and Authentication Methods

Table B-4 lists the caching, filtering, and authentication methods supported by standalone Content Engines that are running the ACNS 5.4.1 software and later releases. An asterisk (*) indicates a feature is supported for that particular protocol.

Table B-4  Caching, Filtering, and Authentication Methods and Related Protocol Support

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Filtering</th>
<th>Proxy Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Caching</td>
<td>N2H2</td>
</tr>
<tr>
<td>HTTP</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>FTP-over-HTTP</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>HTTPS-over-HTTP</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>RTSPG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMS-over-HTTP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTP-WCCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nontransparent native FTP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transparent native FTP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTPS-WCCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTSPG-WCCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMS-over-HTTP-WCCP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported Access Control and Filtering Services for Content Requests

Table B-5 lists the access control and filtering content services that are supported with standalone Content Engines that are running the ACNS 5.4.1 software and later releases. An asterisk (**) indicates a feature is supported for that particular protocol.

Table B-5  Supported Access Control and Filtering Services for Standalone Content Engines

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Access Control</th>
<th>URL Filtering</th>
<th>ICAP</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caching</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>HTTPS</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTP-over-HTTP</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ACNS Software CLI Command Modes for Standalone Content Engines

The ACNS software device mode determines whether the device is functioning as a Content Engine, Content Distribution Manager, Content Router, or IP/TV Program Manager. The commands available from a specific CLI mode are determined by the ACNS software device mode in effect. The default device operation mode is Content Engine.

Table B-6 summarizes the purpose of the different CLI command modes that are available from the Content Engine CLI, which is configured as a standalone Content Engine that is running the ACNS 5.x software. The table also describes how to access the different command modes. The predefined admin account has superuser privileges. By default, the username is admin and the password is default for this predefined admin superuser account. Global configuration commands are device-level commands; subglobal configuration commands are not device-level.

Examples of subglobal configuration modes are the following: interface configuration mode, HTTPS server configuration mode, standard IP ACL configuration mode, and extended IP ACL configuration mode. For more detailed information about these modes, see the Cisco ACNS Software Command Reference, Release 5.5 publication.
<table>
<thead>
<tr>
<th>CLI Command Mode</th>
<th>Purpose</th>
<th>Access</th>
<th>Prompt</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>User EXEC</td>
<td>Monitors the operation of the unit (the standalone Content Engine) and issue some system commands such as telnet, traceroute, and ping.</td>
<td>If you log in using an account that does not have superuser privileges, the following CLI prompt is displayed: ContentEngine&gt; To access user EXEC mode from privileged EXEC mode, enter: ContentEngine# disable where ContentEngine is the hostname of the standalone Content Engine.</td>
<td>ContentEngine&gt;</td>
<td>Use the exit or end command: ContentEngine&gt; exit</td>
</tr>
<tr>
<td>Privileged EXEC</td>
<td>Sets up, monitors, and debugs the standalone Content Engine, including all commands in user EXEC mode.</td>
<td>From user EXEC mode, enter: ContentEngine&gt; enable Can also access privileged EXEC mode by logging into the CLI with an account with superuser privileges.</td>
<td>ContentEngine#</td>
<td>Use the disable command to return to user EXEC mode. ContentEngine# disable</td>
</tr>
<tr>
<td>Global configuration</td>
<td>Configures an ACNS software feature for the entire unit.</td>
<td>From privileged EXEC mode, enter: ContentEngine# configure</td>
<td>ContentEngine (config)#</td>
<td>Use the exit or end command to return to privileged EXEC mode. Alternatively, press Ctrl-Z simultaneously.</td>
</tr>
<tr>
<td>Interface configuration</td>
<td>Configures a particular interface on a standalone Content Engine.</td>
<td>From global configuration mode, use the interface global configuration command. For example, enter: ContentEngine(config)# interface FastEthernet 0/1 ContentEngine(config-if)#</td>
<td>ContentEngine (config-if)#</td>
<td>Use the exit command to return to the previous configuration mode. Use the end command to exit directly to privileged EXEC mode.</td>
</tr>
<tr>
<td>HTTPS server configuration</td>
<td>Configures the HTTPS server on a standalone Content Engine.</td>
<td>From global configuration mode, enter: ContentEngine(config)# https server HTTPS_server_name</td>
<td>ContentEngine (config-https)#</td>
<td>Use the exit command to return to the previous configuration mode. Use the end command to exit directly to privileged EXEC mode.</td>
</tr>
<tr>
<td>Standard IP ACL configuration</td>
<td>Configures standard IP access control lists (ACLs) on a standalone Content Engine.</td>
<td>From global configuration mode, enter: ContentEngine(config)# ip access-list standard acl-name</td>
<td>ContentEngine (config-std-nacl)#</td>
<td>Use the exit command to return to the previous configuration mode. Use the end command to exit directly to privileged EXEC mode.</td>
</tr>
</tbody>
</table>
Appendix B  Reference Material for Standalone Content Engine Deployments

Table B-6  ACNS Software CLI Command Modes for Standalone Content Engines (continued)

<table>
<thead>
<tr>
<th>CLI Command Mode</th>
<th>Purpose</th>
<th>Access</th>
<th>Prompt</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended IP ACL configuration</td>
<td>Configures extended IP ACLs on a standalone</td>
<td>From global configuration mode, enter:</td>
<td>ContentEngine (config-ext-nacl)#</td>
<td>Use the exit command to return to the previous configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Content Engine.</td>
<td>ContentEngine(config-ext-nacl)#</td>
<td></td>
<td>Use the end command to exit directly to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContentEngine(config)## ip access-list extended acl-name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modify acl-num</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>delete Delete a condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>deny Specify packets to reject</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>exit Exit from this submode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>insert Insert a condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>list List conditions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Move Move a condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no Negate a command or set its defaults</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>permit Specify packets to accept</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContentEngine(config-ext-nacl)#</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ACNS Software CLI Online Help and Keyboard Shortcuts

To view the CLI online help, enter a ? as follows:

- View a list of the commands that are available in the current mode:

  ContentEngine(config-ext-nacl)# ?

  delete Delete a condition
  deny Specify packets to reject
  exit Exit from this submode
  insert Insert a condition
  list List conditions
  move Move a condition
  no Negate a command or set its defaults
  permit Specify packets to accept
  ContentEngine(config-ext-nacl)#

- View the available command parameters:

  ContentEngine(config-ext-nacl)# ip access-list extended ?
  (100–199) Extended IP access-list number
  WORD Access-list name (max 30 characters)

- View the possible completions of a partially typed keyword.

To view a description of the online help for the ACNS software CLI, enter the help command.

As a shortcut, you can abbreviate commands to the fewest letters that make them unique. For example, the letters sho can be entered for the show command.

Certain EXEC commands display multiple screens with the following prompt at the bottom of the screen:

```
--More--
```

Press the <Spacebar> to continue the output, or press <Return> to display the next line. Press any other key to return to the prompt. Also, at the "--More--" prompt, you can enter a ? to display the help message.

Table B-7 summarizes the keyboard shortcuts.

Table B-7  Command-Line Processing Keystroke Combinations

<table>
<thead>
<tr>
<th>Keystroke Combinations</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-A</td>
<td>Jumps to the first character of the command line.</td>
</tr>
<tr>
<td>Ctrl-B or the Left Arrow key</td>
<td>Moves the cursor back one character.</td>
</tr>
</tbody>
</table>
Unusable Multicast Address Assignments

The Internet Assigned Numbers Authority (IANA) controls the assignment of IP multicast addresses. The IANA has assigned the IPv4 Class D address space to be used for IP multicast. Therefore, all IP multicast group addresses fall in the range from 224.0.0.0 through 239.255.255.255. However, some combinations of source and group address should not be routed for multicasting purposes. Table B-8 lists the unusable multicast address ranges and the reasons they should not be used.

Some of these addresses have been reserved for use by multicast applications through the IANA. For example, IP address 224.0.1.1 has been reserved for the Network Time Protocol (NTP).

IP addresses reserved for IP multicast are defined in RFC 1112, Host Extensions for IP Multicasting. More information about reserved IP multicast addresses can be found at the following location: http://www.iana.org/assignments/multicast-addresses

You can find all RFCs and Internet Engineering Task Force (IETF) drafts on the IETF website (http://www.ietf.org). The Class D address range is used only for the group address or destination address of IP multicast traffic. The source address for multicast datagrams is always the unicast source address.
## Unusable Multicast Address Assignments

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>224.0.1.2/32</td>
<td>Known insecure service address. See the “Insecure Services” section on page 2-16.</td>
</tr>
<tr>
<td>224.0.1.3/32</td>
<td>Reserved for the discovery of resources within the administrative domain. See the “Limited Scope Addresses” section on page 2-16.</td>
</tr>
<tr>
<td>224.0.1.22/32</td>
<td>Known insecure service address.</td>
</tr>
<tr>
<td>224.0.1.35/32</td>
<td>Reserved for the discovery of resources within the administrative domain.</td>
</tr>
<tr>
<td>224.0.1.39/32</td>
<td>Reserved for the discovery of resources within the administrative domain.</td>
</tr>
<tr>
<td>224.0.1.40/32</td>
<td>Reserved for the discovery of resources within the administrative domain.</td>
</tr>
<tr>
<td>224.0.2.2/32</td>
<td>Known insecure service address.</td>
</tr>
<tr>
<td>224.77.0.0/16</td>
<td>Used to copy files between servers and clients in a local network. See the “Copying Files Between Servers and Clients” section on page 2-16.</td>
</tr>
<tr>
<td>224.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches. See the “Layer 2 Multicast Addresses” section on page 2-17.</td>
</tr>
<tr>
<td>225.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>225.1.2.3/32</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>225.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>226.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>226.77.0.0/16</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>226.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>227.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>227.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>228.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>228.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>229.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>229.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>230.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>230.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>231.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
</tbody>
</table>
### Table B-8  Unusable Multicast Address Assignments (continued)

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>231.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>232.0.0.0/24</td>
<td>Source-specific multicast address. See the “Source-Specific Multicast Addresses” section on page 2-16.</td>
</tr>
<tr>
<td>232.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>233.0.0.0/8</td>
<td>GLOP address. See the “GLOP Addresses” section on page 2-17.</td>
</tr>
<tr>
<td>233.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>234.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>234.42.42.42/32</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>234.142.142/31</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.142.44/30</td>
<td>Used to duplicate files between clients and servers in a local network.</td>
</tr>
<tr>
<td>234.142.48/28</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.142.64/26</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.142.128/29</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.142.136/30</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.142.140/31</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>234.142.142/32</td>
<td>Used to copy files between servers and clients in a local network.</td>
</tr>
<tr>
<td>235.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>235.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>236.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>236.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>236.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>236.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>237.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>237.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
</tbody>
</table>
### Table B-8  Unusable Multicast Address Assignments (continued)

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>238.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>238.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>239.0.0.0/8</td>
<td>Administratively scoped address that should not be passed between administrative domains. See the “Limited Scope Addresses” section on page 2-16.</td>
</tr>
<tr>
<td>239.0.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
<tr>
<td>239.128.0.0/24</td>
<td>Local address that maps to an Ethernet multicast address range and may overwhelm the mapping table of LAN switches.</td>
</tr>
</tbody>
</table>
## Symbols

- wildcard character 7
- .gz extension 44
- .nsc file 39
- .pac files 37

## A

AAA accounting
- activating on Content Engine 5
- command accounting 2
- displaying configuration of 8
- EXEC shell accounting 2
- overview of 2
- statistics for 8
- system accounting 2
- with TACACS+ 2

AAA-based management systems 16

accelerated WCCP support 34

access control
- authentication and authorization
  - of administrative login 1
  - of content requests 2

IP ACLs 2
- RealProxy 8
- access control lists. See IP ACLs

access control services, deploying 15

accessing
- Content Engine CLI 50
- Content Engine GUI 55

access lists, group name-based
- disabling 53

for group authorization 47

access lists, WCCP
- configuring 20
- displaying 20

access-lists 300 command 46

access-lists 300 deny groupname any command 49

access lists enable command 46

ACLs. See IP ACLs

ACNS software CLI
- accessing 51
- command modes 8

ACNS solution
- components of 1
- overview of 1

Active Directory groups
- description of 33
- group searches 49

administrative login accounts
- default account 19
- managing 3
- types of 3

administrative login authentication and authorization
- default 3
- for standalone Content Engines 2

local database
- description of 6
- enabling and disabling 16

overview of 2

RADIUS
- enabling and disabling 17

overview of 6

TACACS+
- enabling and disabling 18
<table>
<thead>
<tr>
<th>Page Reference</th>
<th>Index Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>overview of</td>
</tr>
<tr>
<td>4</td>
<td>administrative login authentication failover</td>
</tr>
<tr>
<td>2</td>
<td>AAA accounting for</td>
</tr>
<tr>
<td>3</td>
<td>login accounts for</td>
</tr>
<tr>
<td>3</td>
<td>administrator levels</td>
</tr>
<tr>
<td>3</td>
<td>normal-level users</td>
</tr>
<tr>
<td>3</td>
<td>superusers</td>
</tr>
<tr>
<td>14</td>
<td>advanced caching features</td>
</tr>
<tr>
<td>15</td>
<td>alarm information</td>
</tr>
<tr>
<td>15</td>
<td>for all alarms</td>
</tr>
<tr>
<td>15</td>
<td>for critical alarms</td>
</tr>
<tr>
<td>15</td>
<td>for major alarms</td>
</tr>
<tr>
<td>15</td>
<td>for minor alarms</td>
</tr>
<tr>
<td>14</td>
<td>alarms</td>
</tr>
<tr>
<td>17</td>
<td>displaying</td>
</tr>
<tr>
<td>15</td>
<td>counts of</td>
</tr>
<tr>
<td>15</td>
<td>history of</td>
</tr>
<tr>
<td>15</td>
<td>list of</td>
</tr>
<tr>
<td>17</td>
<td>status of</td>
</tr>
<tr>
<td>13</td>
<td>overload of</td>
</tr>
<tr>
<td>14</td>
<td>alarm traps</td>
</tr>
<tr>
<td>14</td>
<td>configuring</td>
</tr>
<tr>
<td>14</td>
<td>disabling</td>
</tr>
<tr>
<td>14</td>
<td>enabling</td>
</tr>
<tr>
<td>14</td>
<td>generating</td>
</tr>
<tr>
<td>14</td>
<td>list of</td>
</tr>
<tr>
<td>36</td>
<td>Apache CLF transaction logging format</td>
</tr>
<tr>
<td>36</td>
<td>Apache-style transaction logging</td>
</tr>
<tr>
<td>2</td>
<td>Application and Content Networking System. See ACNS</td>
</tr>
<tr>
<td>54</td>
<td>AUP</td>
</tr>
<tr>
<td>54</td>
<td>configuration of</td>
</tr>
<tr>
<td>54</td>
<td>description of</td>
</tr>
<tr>
<td>12</td>
<td>authenticated HTTP cache settings</td>
</tr>
<tr>
<td>1</td>
<td>authentication</td>
</tr>
<tr>
<td>13</td>
<td>content authentication</td>
</tr>
<tr>
<td>13</td>
<td>end-to-end authentication</td>
</tr>
<tr>
<td>13</td>
<td>HTTP request authentication</td>
</tr>
<tr>
<td>2</td>
<td>traffic bypass</td>
</tr>
<tr>
<td>10</td>
<td>WMT proxy authentication</td>
</tr>
<tr>
<td>12</td>
<td>authentication, authorization, and accounting. See AAA</td>
</tr>
<tr>
<td>12</td>
<td>authentication cache</td>
</tr>
<tr>
<td>12</td>
<td>configuring</td>
</tr>
<tr>
<td>12</td>
<td>size adjustments</td>
</tr>
<tr>
<td>3, 15</td>
<td>authentication configuration command</td>
</tr>
<tr>
<td>17</td>
<td>authentication configuration radius command</td>
</tr>
<tr>
<td>15</td>
<td>authentication configuration tacacs command</td>
</tr>
<tr>
<td>19</td>
<td>authentication databases</td>
</tr>
<tr>
<td>2</td>
<td>standalone Content Engines and</td>
</tr>
<tr>
<td>2</td>
<td>types of</td>
</tr>
<tr>
<td>4</td>
<td>authentication fail-over command</td>
</tr>
<tr>
<td>15</td>
<td>authentication local login command</td>
</tr>
<tr>
<td>17</td>
<td>authentication login local command</td>
</tr>
<tr>
<td>15</td>
<td>authentication login radius command</td>
</tr>
<tr>
<td>19</td>
<td>authentication login tacacs command</td>
</tr>
<tr>
<td>2</td>
<td>authentication servers</td>
</tr>
<tr>
<td>5</td>
<td>LDAP servers</td>
</tr>
<tr>
<td>5</td>
<td>NTLM servers</td>
</tr>
<tr>
<td>5</td>
<td>RADIUS servers</td>
</tr>
<tr>
<td>1</td>
<td>authorization</td>
</tr>
<tr>
<td>1</td>
<td>administrative login authorization</td>
</tr>
<tr>
<td>11</td>
<td>content authorization</td>
</tr>
<tr>
<td>11</td>
<td>bandwidth</td>
</tr>
<tr>
<td>11</td>
<td>and network resources</td>
</tr>
<tr>
<td>11</td>
<td>interface, configuring</td>
</tr>
<tr>
<td>11</td>
<td>bandwidth configuration</td>
</tr>
<tr>
<td>26</td>
<td>advanced configuration</td>
</tr>
<tr>
<td>23</td>
<td>incoming and outgoing</td>
</tr>
<tr>
<td>24, 12</td>
<td>subnet-based bandwidth control</td>
</tr>
<tr>
<td>11</td>
<td>bandwidth control</td>
</tr>
<tr>
<td>11</td>
<td>and content preloading</td>
</tr>
<tr>
<td>2</td>
<td>binary objects, description of</td>
</tr>
<tr>
<td>24</td>
<td>BIOS upgrades</td>
</tr>
<tr>
<td>block.html</td>
<td></td>
</tr>
</tbody>
</table>
customized blocking message 17
browser configuration
for direct proxy routing 5
bypass
authentication traffic 2
static 7
bypass auth-traffic command 5
bypass gateway command 5
bypass lists
displaying summary of 3
static entries in 7
bypass load command 8
bypass options
disabling for URL filtering 12
bypass timer command 4
terminology 8
TFTP and 2
transparent caching 14
caching and streaming services
with direct proxy routing 13
with transparent redirection 14
caching services
configuring with Setup utility 11
conventional caching services 2
RTSP streaming and caching services 5
WMT streaming and caching services 5
calendar
setting 4
caveats
pattern list group type 35
CDP. See Cisco Discovery Protocol
cfs
function of 26, 9
characters
special 34
chunked transfer coding, description of 7
Cisco Discovery Protocol
enabling 2
purpose of 2
Cisco IP/TV, description of 11
Cisco Streaming Engine, description of 10
CiscoWorks2000 26
clear ip access-list counter 24
clear ip wccp command 1
for WCCP Version 2 34
clear statistics rule command 33
clear statistics rule rtp command 33
clear statistics tacacs command 8
clear websense command 27
clear wmt incoming command 45
clear wmt outgoing command 46
clear wmt stream-id command 46
CLF, description of 36
CLI

C

cache
expiration date 10
cache freshness
for FTP cache 40, 43
for HTTP cache 9
cache hits
description of 2
refresh 13

cache misses
description of 2
no-proxy action 13
using upstream proxy 13

caching
CTE HTTP objects 7
description of 2
disabling for certain clients 24
hierarchy 10
HTTP and 2
network protocols and 2
nontransparent caching
proxy-style 13
accessing 51
command modes 8
EXEC mode 9
global configuration mode 9
HTTPS server configuration mode 9
interface configuration mode 9
keyboard shortcuts 10
logging in to
  accessing 50
  usage guidelines 10
clients
  description of 3
  support of 12
clock
clearing and setting 4
daylight saving and local time, setting 4
displaying standard timezones 4
UTC offset, setting 4
clock summertime command 4
command accounting
  activating on Content Engine 2
  overview of 4
command-line interface. See CLI
configuring
AAA accounting 1
administrative login authentication and authorization 8
alarm traps 14
authentication and authorization
  for administrative users 8
  for content requests 1
bandwidth 11
content preloading 2
conventional caching services 2
custom-web-cache service
  on Content Engine 20
disk space allocation 25, 9
dns-cache service
  on Content Engines 62
FTP connection settings 36
FTP native caching service
  on Content Engines 55
HTTP request authentication 16
https-cache service
  on Content Engines 27
  on WCCP routers 29
ICAP services 4
interface bandwidth 11
IP ACLs 10
IP spoofing 10
native ftp caching service
  on WCCP routers 29
passwords 24
playserver 10
RADIUS authentication 10
reverse-proxy service
  on Content Engines 23
  on WCCP routers 33
Rules Template 1
standard web-cache service
  on Content Engines 18
  on WCCP routers 27
static bypass 7
transaction logging 1
URL filtering 1
user-defined WCCP services 15
content
  classifications 5
  on-demand 5
  preloaded 5
  pre-positioned 6
types of 5
content authentication
  of HTTP requests 13
  overview of 2
content authorization
  of LDAP users 47
  of NTLM users 48
overview of 2
content distribution, types of 5
Content Engine
  ACNS software GUI
    logging out 57
    main components of 53
    automatic reload 18
    bandwidth configuration 11
    configuring for IP spoofing 10
    configuring for WCCP 11
    mounting to a NAS device 13
    rebooting 24
    removing or replacing 24
    security 35
Content Engine administrators
  normal-level users 3
  superusers 3
Content Engine CLI
  command modes 8
  logging in to 50
  logging out of 52
  online help 10
Content Engine clusters
  configuring ICP clients for 5
  configuring ICP servers for 5
Content Engine GUI
  disabling access to 54
  enabling access to 54
  exiting 57
  feature tabs 2
  logging in to 55
  logging out from 57
  managing administrative login accounts 8
  Reporting tab and subtabs 7
  System tab and subtabs 6
  WCCP tab and subtabs 2
content preloading
  configuring 2
  creating filter for 5
  creating preload URL list file 3
  default start time 5
  enabling 2
  forcing 5
  NTLM preloading support 2
  preloading authenticated content 3
  resuming after aborting 11
  scheduling 5
  setting DSCP 6
  setting Type of Service (ToS) 6
  specifying maximum bandwidth for
    stopping operation 11
  viewing statistics for 8
  viewing statistics for current preload 8
  viewing status of current preload 7
conventional caching services
  configuring 2
conventions, documentation 26
copy running-config command 13
CTE HTTP objects, caching 7
custom blocking messages
  in URL filtering 12
custom format transaction logging 36
custom messages
  for FTP proxy responses 19
  for HTTP requests 14
  custom web cache service
    configuring
      on Content Engines 20
      on WCCP routers 32

D
daylight savings time
  setting 4
daylight saving time
  setting 4
db e a d ip wccp events command 42
debug ip wccp packets command
**Index**

- **for WCCP Version 1** 42
- **debug wmt error command** 55
- **debug wmt trace command** 55
- **default administrative login account** 19
- **default user account** 3
- **directing requests back to proxy server** 7
- **direct proxy routing**
  - caching services with 13
  - configuring
    - browsers for 5
    - media players for 5
    - forward proxy caching and 3
    - streaming services with 13
    - supported services for 13
- **disabling**
  - access to Content Engine GUI 54
  - caching for certain clients 24
  - local authentication and authorization 16
  - persistent connections 68
  - RADIUS authentication 20
  - RADIUS authentication and authorization 17
  - RealProxy caching 30
  - SNMP agents 10
  - TACACS+ authentication and authorization 18
  - transparent caching services 14
  - WCCP flow redirection 17
- **disk add command** 9
- **disk config command** 11
- **disk config sysfs command** 10
- **disk configuration, displaying** 12
- **disk configuration settings**
  - configuring
    - through CLI 14
    - through Setup utility 14
- **disk delete-partitions command** 11
- **disk drives**
  - error handling thresholds
    - description of 18
    - specifying 18
  - manually unmarking 19
  - removing partitions from 11
  - disk error-handling reload command 18
  - disk error-handling threshold command 18
  - disk mark command 19
  - disk partitions, removing 11
  - disks
    - configuring
      - space allocation 25, 9
      - sysfs space 11
    - NFS and CIFS storage 18, 13
    - storage types 9
    - disk use command
      - disk drives
        - unusing 12
    - dns-cache service
      - configuring
        - on routers 28
        - on standalone Content Engines 62
    - DNS caching service
      - configuring WCCP transparent interception for 64
      - disabling 67
      - overview of 63
      - dns use-original-server command 64
    - documentation conventions 26
    - DSCP
      - rule action 12
      - Rules Template 6
  - dynamic traffic bypass 5

**E**

- **ecn enable command** 3
- **EIM**
  - through content authentication 2
  - through group-based authorization 11
  - through URL filtering with SmartFilter 39
- **employee Internet management. See EIM**
- **enabling 2**
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamic authentication bypass</td>
<td>4</td>
</tr>
<tr>
<td>local authentication and authorization</td>
<td>16</td>
</tr>
<tr>
<td>persistent connections</td>
<td>68</td>
</tr>
<tr>
<td>RADIUS authentication and authorization</td>
<td>17</td>
</tr>
<tr>
<td>SNMP agent</td>
<td>6</td>
</tr>
<tr>
<td>transparent error handling</td>
<td>5</td>
</tr>
<tr>
<td>WCCP</td>
<td>12</td>
</tr>
<tr>
<td>WCCP flow redirection</td>
<td>17</td>
</tr>
<tr>
<td>WCCP on routers</td>
<td>23</td>
</tr>
<tr>
<td>enabling and disabling</td>
<td></td>
</tr>
<tr>
<td>NTLM allow domain lists</td>
<td>44</td>
</tr>
<tr>
<td>error logging, WMT</td>
<td>11</td>
</tr>
<tr>
<td>EtherChannel configuration</td>
<td>2</td>
</tr>
<tr>
<td>Ethernet MAC address mapping</td>
<td>17</td>
</tr>
<tr>
<td>EXEC CLI mode</td>
<td>9</td>
</tr>
<tr>
<td>EXEC shell accounting</td>
<td></td>
</tr>
<tr>
<td>activating on Content Engine</td>
<td>2</td>
</tr>
<tr>
<td>overview of</td>
<td>3</td>
</tr>
<tr>
<td>extended IP ACLs</td>
<td></td>
</tr>
<tr>
<td>configuration examples</td>
<td>22</td>
</tr>
<tr>
<td>configuring</td>
<td>10</td>
</tr>
<tr>
<td>supported conditions</td>
<td>14</td>
</tr>
<tr>
<td>supported keywords</td>
<td>12</td>
</tr>
<tr>
<td>TCP keywords</td>
<td>12</td>
</tr>
<tr>
<td>typical uses of</td>
<td>5</td>
</tr>
<tr>
<td>Extended Squid-style transaction logging</td>
<td>35</td>
</tr>
<tr>
<td>Extended Squid transaction log format</td>
<td>36</td>
</tr>
<tr>
<td>external FTP server</td>
<td></td>
</tr>
<tr>
<td>exporting transaction logs to</td>
<td>43</td>
</tr>
<tr>
<td>permanent error from</td>
<td>45</td>
</tr>
<tr>
<td>Fibre Channel interface</td>
<td></td>
</tr>
<tr>
<td>configuring</td>
<td>9</td>
</tr>
<tr>
<td>file system types</td>
<td></td>
</tr>
<tr>
<td>description (table)</td>
<td>9</td>
</tr>
<tr>
<td>filtering services</td>
<td></td>
</tr>
<tr>
<td>deploying</td>
<td>15</td>
</tr>
<tr>
<td>URL filtering</td>
<td>11</td>
</tr>
<tr>
<td>flow protection</td>
<td></td>
</tr>
<tr>
<td>WCCP</td>
<td>9</td>
</tr>
<tr>
<td>forwarding</td>
<td></td>
</tr>
<tr>
<td>HTTP, HTTPS, and FTP proxy-style requests</td>
<td>7</td>
</tr>
<tr>
<td>forward proxy caching, overview of</td>
<td>3</td>
</tr>
<tr>
<td>FTP</td>
<td></td>
</tr>
<tr>
<td>configuring connection settings</td>
<td>38</td>
</tr>
<tr>
<td>description of</td>
<td>2</td>
</tr>
<tr>
<td>FTP native caching</td>
<td></td>
</tr>
<tr>
<td>overview of</td>
<td>54</td>
</tr>
<tr>
<td>FTP caching</td>
<td></td>
</tr>
<tr>
<td>nontransparent FTP native caching</td>
<td>42</td>
</tr>
<tr>
<td>transparent FTP native caching</td>
<td>54</td>
</tr>
<tr>
<td>ftp min-ttl command</td>
<td>10</td>
</tr>
<tr>
<td>FTP native caching</td>
<td></td>
</tr>
<tr>
<td>nontransparent caching</td>
<td></td>
</tr>
<tr>
<td>caching</td>
<td>42</td>
</tr>
<tr>
<td>overview of</td>
<td>54</td>
</tr>
<tr>
<td>transparent caching</td>
<td></td>
</tr>
<tr>
<td>configuring</td>
<td>55</td>
</tr>
<tr>
<td>displaying statistics for</td>
<td>46, 58</td>
</tr>
<tr>
<td>ftp native caching service</td>
<td></td>
</tr>
<tr>
<td>configuring on standalone Content Engines</td>
<td>55</td>
</tr>
<tr>
<td>ftp-native custom-message command</td>
<td>19</td>
</tr>
<tr>
<td>ftp-native proxy active-mode command</td>
<td>55</td>
</tr>
<tr>
<td>FTP native proxy servers</td>
<td></td>
</tr>
<tr>
<td>displaying configuration of</td>
<td>57</td>
</tr>
<tr>
<td>FTP objects</td>
<td></td>
</tr>
<tr>
<td>configuring object freshness</td>
<td>4</td>
</tr>
<tr>
<td>FTP-over_HTTP proxy servers</td>
<td></td>
</tr>
<tr>
<td>displaying configuration of</td>
<td>41</td>
</tr>
<tr>
<td>displaying statistics for</td>
<td>41</td>
</tr>
<tr>
<td>Fast Cache</td>
<td></td>
</tr>
<tr>
<td>configuring on Content Engines</td>
<td>31</td>
</tr>
<tr>
<td>Fast Start</td>
<td></td>
</tr>
<tr>
<td>configuring on Content Engines</td>
<td>29</td>
</tr>
</tbody>
</table>
FTP-over-HTTP caching
  configuring  39
ftp-over-http object max-size command  40
ftp-over-http proxy active-mode enable command  40
ftp-over-http proxy anonymous-pswd command  40
FTP-over-HTTP proxy caching
  specifying incoming ports  39, 43
ftp-over-http proxy incoming command  39, 43
ftp-over-http proxy outgoing command  40
ftp-over-http proxy outgoing monitor command  8
ftp-over-http reval-each-request all command  40
ftp proxy outgoing host command  4
FTP requests
  native
    configuring access lists for  42, 19
    configuring nontransparent FTP native caching  42
    configuring proxy authentication for nontransparent requests  55
    configuring transparent FTP native caching  54
    creating custom messages for  19
ftp-server gw proto command  60
FTP servers
  exporting transaction logs to  43
FTP statistics, displaying  28

G

generic routing encapsulation. See GRE encapsulation
global configuration CLI mode  9
global exclusion from proxy forwarding  7
GLOP addresses  17
GRE encapsulation  5
group authorization
  Active Directory group searches  49
    methods of  2
    of LDAP users  47
GUI. See Content Engine GUI
gui-server command  54
gzip format
to compress archived log files  44

H

healing mode, configuration of  70
hierarchical caching  10
hostname command  14
HTTP  14
  authenticated cache settings  12
  cache freshness settings  9
  outgoing proxy exclusion  5
  proxy failover  2
  status codes of  37
http age-multiplier command  10, 11
http append x-forwarded-for-header command  15, 11
http authenticate-strip-ntlm command  8
HTTP authentication cache
  configuring  12
  displaying configuration of  16
HTTP authentication cache, configuring  12
http authentication cache timeout command  17
http authentication command  12
http cache-authenticated ntlm command  2, 3
http cache-chunk-encoded enable command  7
http cache-cookies command  10
HTTP caching
  configuring
    nontransparent forward proxy caching  8
    reverse proxy caching  23
    transparent HTTP forward proxy caching  17
  types of  7
http cluster command  71
http cluster heal-port command  71
http cluster http-port command  71
http cluster max-delay command  71
http cluster misses command  72
http custom-error-page command  16
http l4 switch enable command  5
http max-ttl command  11
http min-ttl command 10, 11
http monitor url command 5
HTTP objects
  configuring object freshness 4
HTTP proxy caching
  configuring
    through Content Engine CLI 7
    through Setup utility 29
http proxy incoming command 14
http proxy outgoing host command 3
http proxy outgoing monitor command 8
http proxy outgoing origin-server command 2
HTTP request authentication 44
  configuring 16
  hierarchical caching
    in proxy server mode 14
  logging failures of 50
  methods of 2
  overview of 13
  transaction logging 18
  transparent mode 15
HTTP requests
  creating custom message pages for 14
http reval-each-request all command 11
http reval-each-request command 11
HTTP reverse proxy caching
  configuring
    through Content Engine CLI 14
    through Setup utility 14
HTTPS
  description of 3
  Rules Template with 2
  server configuration CLI mode 9
  statistics, displaying 28
https-cache service
  configuring
    on sContent Engines 27
    on WCCP routers 29
HTTPS caching
  configuration of 24
  SSL termination and 24
https certgroup command 33
https command 32
HTTPS errors, displaying 28
HTTPS outgoing proxy servers
  configuring 27
  displaying current state of 27
https proxy command 26
HTTPS proxy commands 26
https proxy outgoing host command 34, 4
https proxy outgoing monitor command 8
HTTPS statistics, displaying 28
HTTP statistics, displaying 28
HTTP transparent caching
  configuring
    through Content Engine CLI 14
    through Setup utility 14
IANA 16, 11
ICAP 5
  approved vendors 3
  configuring
    Content Engine settings 5
    server 8
    services 7
  maximum file size supported 4
  overview of 1
icap append-x-headers command 5
icap apply command 4
icap bypass streaming-media command 6
icap logging enable command 4
icap service command 7
ICAP services
  configuring 4
  description of 2
icap service server command 8
ICP
  client settings 73
  server settings 74
  icp client command 73
ICP clients
  configuring 5
  icp server command 74
ICP server configuration 74
IFP
  role of 19
  inetd enable tftp command 59, 60
  interface bandwidth 11
  interface command 19, 2
  interface configuration CLI mode 9
  interface-level DHCP 7
Internet Assigned Numbers Authority. See IANA
Internet Cache Protocol. See ICP
Internet Content Adaption Protocol. See ICAP
Internet Filtering Protocol. See IFP
Internet service provider. See ISP
ip access-group command 16
ip access-list command 3
ip access-list extended command 5
ip access-list standard command 5
IP ACLs
  activating 6
  activating on an interface 16
  applying to interface 17
  clearing counter 24
  clearing IP ACL counter 24
  configuration modes 9
  defining 6
  deleting 23
  description of 2
  extended configuration mode
    accessing 5
    description of 5
  extended IP ACLs
    typical uses of 5
  naming guidelines 9
  overview of
    standard configuration mode
      accessing 5
      description of 5
    standard IP ACLs
      typical uses of 5
      types of 4
  viewing configuration of 23
ip address dhcp command 17
IP addresses
  multiple IP addresses on a single interface 9
  ip default-gateway command 14
  ip default-gateway command (global configuration) 5
  ip domain-name command 14
IP multicasting
  and insecure services 16
  copying files between server and client 16
  Ethernet MAC address mapping 17
  fundamentals 16
  GLOP addresses 17
  Layer 2 multicast address considerations 17
  limited scope addresses 16
  Protocol Independent Multicast. See PIM
  source-specific multicast addresses 16
  ip name-server command 64
  ip name-servers command 14
  ip route-cache same-interface command 24
  ip route command 5
IP spoofing
  configuration of 10
  examples of 12
  WCCP and 10
  ip wccp command 26
  ip wccp password command
    WCCP Version 2 commands
      ip wccp password 23
  ip wccp redirect-list command 26
  ip wccp redirect out command 25
ip web-cache redirect command  1, 26

K
keystroke combinations, CLI  10

L
launching
   Setup utility  19
Layer 2
   multicast address  17
   redirection  8
Layer 4 switches
   forward proxy caching and  8
   reverse proxy caching and  13
LDAP
   security options  23
   LDAP acceptable use policy
      configuration of  54
      description of  54
   LDAP Active Directory  33
   LDAP databases
      contents of  28
      entries in  29
      querying  33
      structure of  28
   LDAP directory service
      description of  28
      directory entries  28
   LDAP memory cache
      disabling  35
      specifying the maximum TTL for an object  53
      specifying the size of  35
   LDAP nested static groups
      description of  31
      searching for user account information  37
   LDAP password expiration
      configuration of  54
      description of  54
   ldap server command  23
   ldap server enable command
      LDAP authentication
         for HTTP requests  23
   ldap server group command  24
   ldap server password-expiry command  54
   ldap server policy-redirect attribute command  54
   ldap server policy-redirect enable command  54
   ldap server policy-redirect redirect-url command  54
   LDAP servers
      configuring for Content Engine  5
      types of  22
   ldap server version 3 command  33
   LDAP Version 3
      enabling  33
   less command  9
   license keys
      RealProxy
         restoring factory defaults  29
         uninstalling  31
   limited scope addresses  16
   list of
      nontransparent mode services  13
      supported caching and streaming services
         with transparent redirection  14
      WCCP features and services  10
      WCCP services  3
   live content
      description of  6
   live splitting
      description of  10
   load-balance command  4
   load balancing  5
   load shedding  5
   log files
      exporting  43
      restarting export of  45
logging command 22
logging console priority command 23
logging disk command 24
logging host command 25
logging host hostname command 25
logging in to
  Content Engine GUI 55
logging on
to Content Engine
  using GUI 50
  using SSH 52
  using Telnet 50
logging out
  of Content Engine CLI 52
  of Content Engine GUI 57
login authentication
  local method 15

M
MAC address mapping 17
MAC address table 17
mapping of RealProxy error level to syslog priority level 26
media file system. See mediafs
mediafs
  configuring
    through the Setup utility 25
  function of 9
media player configuration
  for direct proxy routing 5
media player configuring
  for direct proxy routing 5
menu options
  Setup utility 14
MIBs
  supported 5
  monitoring
    with ACNS software alarms 11
    with SNMP 2
    with the Cisco Discovery Protocol 2
  monitoring URLs 5
mounting to NAS devices 18, 13
multicast
  WMT, logging 48
multicast address
  unusable assignments 12
multicasting
  See also IP multicasting
multicast stations
  defining 40
  description of 40
  starting and stopping 42
multiple IP addresses
  configuring on a single interface 9
multiple router support 8

N
N2H2 URL filtering 12
NAS devices
  mounting to 18, 13
native ftp caching service
  configuring
    on WCCP routers 29
near-miss
  definition 70
network connectivity
  testing 10
network interfaces
  configuring 2
  configuring
    through CLI 14, 19
no authentication login local enable command 17
no authentication login radius enable command 18
no auto-register enable command 22
Node Health Manager
description of 11
Node Manager
description of 13
no disk error-handling reload command 18
no dns-cache size command 67
no ntlm basic-auth enable command 45
no ntlm server ad-group-search mem-cache command 35
nontransparent caching
  SSL tunneling and 6
nontransparent FTP native caching
description of 42
nontransparent mode
  forward proxy caching 3
nontransparent mode services
  list of 13
nontransparent requests
description of 4
no pre-load enable command 11
no radius-server enable command
  RADIUS authentication
    disabling 20
no rtsp proxy media-real enable command 30, 31
no snmp-server command 10
no snmp-server enable traps command 11
no transaction-logs export enable command 47
no transaction-logs logging enable command 48
no transaction-logs logging facility command 48
no transaction-logs logging host command 48
no wccp version command 14
no wmt enable command 31, 13
no wmt license-key command 13
NTLM
description of 3
end-to-end authentication
  object caching 9
  pass-through service 9
NTLMv1
  re-enabling 6
NTLMv2
  enabling 6
ntlm allow-domain enable command 44
NTLM allow domain lists
  enabling and disabling 44
ntlm basic-auth enable command 45
NTLM preloading support 2
ntlm server ad-group-search command 52
ntlm server ad-group-search mem-cache command 52
ntlm server ad-group-search mem-cache max-ttl command 53
ntlm server ad-group-search mem-cache size command 53
ntlm server command 41
ntlm server enable command 49
ntlm server host command 48
NTLM servers
  configuring for Content Engine 5
ntlm version 2 command 6

O
on-demand content
description of 5
origin web servers
  description of 3
outgoing FTP-over-HTTP proxy servers
  monitoring 8
outgoing FTP proxy servers
  configuration of 8
  configuring 4
  designating primary server 3
outgoing HTTP proxy servers
  designating primary server 3
  monitoring 7
outgoing HTTPS proxy servers
  configuration of 8
  configuring 4
  designating primary server 4
  monitoring 8
outgoing proxy servers
  primary outgoing HTTP proxy server 3
  primary outgoing HTTPS proxy server 4
overload bypass
  configuring 8
  description of 2
  overview of 8

P

packet flow
  diagram 2
packet marking 12
PAP
  password authentication 12
Password Authentication Protocol. See PAP
password configuration 24
pattern list group type
  caveats 35
pattern lists
  adding pattern to 29
  associating an action with 30
  configuring 28
persistent connections
  configuring 5
  disabling 68
  enabling 68
  purpose of 68
PIM 16
ping command 10
PNA
  support of 3
port lists
  WCCP services and 12
pre-load concurrent-requests command 4
pre-load depth-level-default command 4
pre-load dscp command 6
preloaded content 5
pre-load enable command 4
pre-load fetch domain command 5
pre-load force command 5
preloading authenticated content 3
preloading content
  bandwidth control 11
  support for ToS and DSCP 6
pre-load max-bandwidth command 5
pre-load no-fetch suffix command 5
pre-load schedule command 5
pre-load schedule every-week command 6
pre-load traverse-other-domains command 5
pre-load url-list-file command
  creating 3
  description of 3
pre-load url-list-file command 3, 5
pre-positioned content, description of 6
primary interface
  changing to a different interface 8
primary outgoing proxy servers
  designating for FTP-over-HTTP 4
  designating for HTTP 3
  designating for HTTPS-over-HTTP 27, 4
Progressive Networks Audio. See PNA
Protocol-Independent Multicast. See PIM
proxy authentication
  WMT caching proxy 10
proxy authentications
  for nontransparent FTP native requests 55
  types of 10
proxy-auto-config download command 38
proxy-auto-config enable command 38
proxy failover 2
proxy forwarding
  excluding items 6
proxying services
  overview of 7
proxy protocols
  parameters 6
proxy-protocols command 6
proxy-protocols outgoing-proxy exclude command 34
proxy-protocols transparent command 34
proxy-protocols transparent default-server command 34
proxy-protocols transparent original-proxy command 26
proxy-protocols transparent reset command 34
proxy servers
description of 7
proxy-style requests, description of 4

Q
QuickTime, description of 10

R
RADIUS
URL filtering 12
RADIUS authentication
counting 10
overview 6
RADIUS key
specifying on Content Engine 10
radius-server command 16
radius-server host command 11
radius-server key command 11
radius-server retransmit command 11
RADIUS servers
as authentication servers 10
authentication settings 10
configuring for Content Engine 5
redirect 13
radius-server timeout command 11
Range requests 67
RealMedia
description of 10
Real Media caching
counting through Setup utility 15
RealMedia players, description of 13
Real Media proxy caching
counting through Content Engine CLI 15
through Setup utility 15
Real Media transparent caching
configuring through Content Engine CLI 15
RealPlayer 46
RealProxy
and access control 8
disabling 13
disabling RealProxy caching 30
enabling 13
error codes 26
features and benefits 3
license keys
restoring factory defaults 29
licenses
accepting 13
evaluation 13
permanent 13
specifying 13
restarting 30
restoring default configuration 29
RealProxy caching
disabling 30
RealProxy license key
installing 13
uninstalling 31
RealProxy statistics
displaying 31
RealSystem administrator page 22
RealTime Streaming Protocol. See RTSP
rebooting Content Engine 24
redirected HTTP traffic 20
registering router
to a multicast address 24
regular expressions
special characters 34
reloading Content Engine
    automatic reload option 18
removing
    Content Engines 24
disk partitions 11
replacing Content Engines 24
restarting export of log files 45
retransmit count
    for RADIUS server authentication 10
    for TACACS+ server authentication 12
reverse proxy caching
    Layer 4 CSS switch
        configuration process 14
        through Layer 4 switching 13
        through WCCP-enabled routers 10
reverse proxy service
    configuring on Content Engine 2
reverse-proxy service
    configuring
        on Content Engines 23
        on routers 33
        on WCCP routers 33
router lists
    defining for WCCP services 12
RTSP
    description of 3
    overview of 3
    Rules Template 2
    support of 3
RTSP gateway
    overview 10
    purpose of 14
    related commands 16
rtsp ip-address command 16
rtsp l4-switch enable command 18
rtsp proxy media-real accept-license-agreement command 13
rtsp proxy media-real enable command 13
RealMedia proxy caching
    enabling 20
rtsp proxy media-real evaluate command 13
rtsp real-proxy default-configuration command 14
rtsp real-proxy restore factory-default command 29
RTSP streaming and caching services
    description of 5
    list of 5
rule action no-persistent-connection command 68
rule actions
    list of 11
    order of execution 20
    overview of 5
    supported action and pattern combinations 17
rule command 20
rule no-auth domain command 17
rules processing
    configuring 23
deleting rules 31
    enabling 23
    performance guidelines 35
Rules Template
    actions
        associating with a pattern list 30
        list of 11
        order of execution 20
    configuring 1
deleting rules 31
DSCP 6
    enabling rule processing 23
    order of executing actions 20
pattern lists
    adding patterns to 29
    configuring 28
    patterns
        list of 7
        order of execution 21
    setting ToS or DSCP 12
    supported rule patterns 7
S
sanitizing transaction logs
   enabling 42
   purpose of 42
satellite connections
   and TCP 6
scheduling
   content preloading 5
secret keys
   RADIUS 10
   TACACS+ 12
Secure Shell. See SSH
security on Content Engine 35
setting password 24
setup command 19
Setup utility
   configuring
      HTTP proxy caching 29
      HTTP reverse proxy caching 14
      HTTP transparent caching 14
      Real Media proxy caching 15
      Real Media transparent caching 15
      WMT proxy caching 15
      WMT transparent caching 15
disk configuration 14
   keys and arrows 15
   launching 19
   manually invoking 12
   menu options 14
   usage guidelines 12
sfadmin console 39
show alarm command 15
show alarm critical command 15
show alarm detail command 15
show alarm major command 15
show alarm minor command 15
show alarms command 15
show alarms history command 15
show alarms history detail support command 16
show alarms status command 17
show authentication user command
   administrative authentication
      displaying configuration of 19
show bypass summary command 3
show clock command 17
show clock standard-timezones details command 5
show clock standard-timezones region command 5
show clock standard-timezones zones command 5
show clock standard-timezones all command 4
show disks command 10
show disks details command 12
show ftp-native command 57
show ftp-over-http command 41
show ftp-over-http statistics command 41
show ftp proxy command 8
show http authentication command 16
show http proxy command 2, 8
show https proxy command 27, 8
show http ttl command 12
show inventory command 2
show ip interface command
   for WCCP Version 1 39, 41
show ip wccp web-caches command
   for WCCP Version 1 42
show logging command
   syslog hosts
      displaying configuration for 22
show pre-load command 7
show rtsp proxy media-real license-agreement command 13
show running-config command 4
show snmp group command 10
show statistics bandwidth advanced command 26
show statistics ftp-native command 46, 58
show statistics http command 8
show statistics http miss-reason command 3
show statistics http performance command 2
show statistics http requests command 48
<table>
<thead>
<tr>
<th>Command/Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>show statistics http savings command</td>
<td>2</td>
</tr>
<tr>
<td>show statistics pre-load command</td>
<td>8</td>
</tr>
<tr>
<td>show statistics rtsp proxy command</td>
<td>31</td>
</tr>
<tr>
<td>show statistics rtsp proxy media-real requests command</td>
<td></td>
</tr>
<tr>
<td>show statistics url-filter http websense command</td>
<td>38</td>
</tr>
<tr>
<td>show statistics wccp gre command</td>
<td>22</td>
</tr>
<tr>
<td>show statistics wmt requests command</td>
<td>47</td>
</tr>
<tr>
<td>show statistics wmt streamstat command</td>
<td>47</td>
</tr>
<tr>
<td>show tech-support command</td>
<td>2</td>
</tr>
<tr>
<td>show transaction-log command</td>
<td>51</td>
</tr>
<tr>
<td>show wccp command</td>
<td>41</td>
</tr>
<tr>
<td>for WCCP Version 1</td>
<td>42</td>
</tr>
<tr>
<td>for WCCP Version 2</td>
<td>17</td>
</tr>
<tr>
<td>show wccp content-engines command</td>
<td>17</td>
</tr>
<tr>
<td>show wccp flows ftp-native command</td>
<td>9</td>
</tr>
<tr>
<td>show wccp gre command</td>
<td>49</td>
</tr>
<tr>
<td>show wccp masks command</td>
<td>37</td>
</tr>
<tr>
<td>show wccp masks ftp-native command</td>
<td>57</td>
</tr>
<tr>
<td>show wccp services command</td>
<td>16</td>
</tr>
<tr>
<td>show wmt diagnostics command</td>
<td>11</td>
</tr>
<tr>
<td>show wmt http allow extension command</td>
<td>21</td>
</tr>
<tr>
<td>show wmt license-agreement command</td>
<td>17</td>
</tr>
<tr>
<td>shutting down WCCP</td>
<td>17</td>
</tr>
<tr>
<td>SNMP traps</td>
<td></td>
</tr>
<tr>
<td>configuring</td>
<td>8</td>
</tr>
<tr>
<td>disabling</td>
<td>11</td>
</tr>
<tr>
<td>enabling</td>
<td>9</td>
</tr>
<tr>
<td>source-specific multicast address</td>
<td>16</td>
</tr>
<tr>
<td>special characters</td>
<td></td>
</tr>
<tr>
<td>in regular expressions</td>
<td>34</td>
</tr>
<tr>
<td>Squid transaction log format</td>
<td>36</td>
</tr>
<tr>
<td>SSH, support of</td>
<td>52</td>
</tr>
<tr>
<td>sshd command</td>
<td>52</td>
</tr>
<tr>
<td>ssh-key-generate command</td>
<td>52</td>
</tr>
<tr>
<td>SSL tunneling</td>
<td></td>
</tr>
<tr>
<td>nontransparent caching and</td>
<td>6</td>
</tr>
<tr>
<td>standard IP ACLs</td>
<td></td>
</tr>
<tr>
<td>configuring</td>
<td>10</td>
</tr>
<tr>
<td>supported conditions</td>
<td>14</td>
</tr>
<tr>
<td>typical uses of</td>
<td>5</td>
</tr>
<tr>
<td>standard timezones, displaying</td>
<td>4</td>
</tr>
<tr>
<td>standard web-cache service</td>
<td></td>
</tr>
<tr>
<td>configuring</td>
<td></td>
</tr>
<tr>
<td>on standalone Content Engines</td>
<td>18</td>
</tr>
<tr>
<td>on WCCP routers</td>
<td>27</td>
</tr>
<tr>
<td>standby groups, interfaces of</td>
<td>5</td>
</tr>
<tr>
<td>starting and stopping</td>
<td></td>
</tr>
<tr>
<td>WMT multicast stations</td>
<td>41</td>
</tr>
<tr>
<td>startup dialog</td>
<td></td>
</tr>
</tbody>
</table>
for configuring network settings 21
static bypass
  configuring 7
  examples 7
statistics
  general CFS disk 7
  performance 7
  TCP requests 7
  WMT streaming 7
statistics for 8
statistics wmt requests command 47
streaming media
  sample deployment 12
  supported protocols 12, 3
streaming media caching
  description of 11
streaming media files
  delivery of 11
  description of 11
streaming media players
  Cisco IP/TV Viewer 13
  QuickTime 13
  RealMedia players 13
streaming media protocols
  RTSP 3
streaming media services
  configuring 11
  licenses for 10
streaming media solutions
  Cisco IP/TV 11
  Cisco Streaming Engine 10
  QuickTime 10
  RealMedia 10
  WMT 9
streaming services
  supported with
    direct proxy routing 13
    transparent redirection 14
  stream splitting 39
superuser
  AAA accounting for 2
  description of 19
sysfs
  configuring disk space for 11
  purpose of 26, 9
syslog
  priority level mapping to RealProxy error codes 27, 26
syslog hosts, configuring 24
system accounting
  activating on Content Engine 2
  overview of 3
system logging
  configuring 25
    to console 23
    to disk 24
    to remote hosts 24

T

TACACS 18
TACACS+
  AAA accounting and 2
TACACS+ authentication
    configuring for HTTP requests 20
    password authentication method 12
TACACS+ authentication and authorization
    enabling 18
    overview of 7
TACACS+ enable password attribute 8
TACACS+ key
    specifying on Content Engine 12
TACACS+ servers
    specifying as authentication servers 12
tacacs key command 12
tacacs password command 13
tacacs retransmit command 13
tacacs server command 12
tacacs timeout command 13
TCP
  configuring parameters using Content Engine GUI 2
  over satellite extensions 6
tcp client-mss command 3
tcp client-receive-buffer command 3
tcp client-rw-timeout command 3
tcp client-satellite command 3
tcp client-send-buffer command 3
tcp cwnd-base command 3
tcp increase-xmit-timer-value command 3
tcp init-ss-threshold command 3
tcp keepalive-probe-cnt command 3
tcp keepalive-probe-interval command 3
tcp keepalive-timeout command 3
TCP memory
  configuring limits for 4
tcp memory-limit command 4
TCP memory limits
  default settings for 5
tcp server-mss command 3
tcp server-receive-buffer command 3
tcp server-rw-timeout command 3
tcp server-satellite command 3
tcp server-send-buffer command 2
telnet command 10
testing
  connectivity of URLs
    for FTP-over-HTTP 8
    for HTTP 7
test-url command
troubleshooting
  with the HTTP CLI client 7
test-url ftp test-URL command 8
test-url http test-URL command 7
text conventions 26
text objects
  description of 2
TFTP
  caching and 2
  description of 2
TFTP gateway
  configuring 60
  enabling 58, 59
TFTP server
  configuring 60
  enabling 58, 59
tftp-server access-list command 61
tftp-server dir command 59, 60
  thresholds
    disk error handling 18
  timeout intervals
    for RADIUS server authentication 10
    for TACACS+ server authentication 12
timeout of a nonresponsive host 10
time zone setting 4
token strings 36
ToS
  Rules Template 12
  setting 12
trace-route command 11
transaction logging
  and NTLM authentication 42
  Apache-style format 36
custom format 36
default settings 32
  Extended Squid-style format 35
  for HTTP request authentication 18
  formats 36
  specifying facility 49
WMT 10, 48
transaction-log logging entry-type all command 50
transaction logs
  archiving of 46
  displaying configuration of 51
  exporting 43
  including domain name in 49
  including username in 49
  sanitizing 42
sending to remote syslog hosts 49
specifying entry type 50
transaction-logs archive command 46
transaction-logs export sftp-server command 44
transaction-logs format apache command 36
transaction-logs format custom command 36
transaction-logs format extended-squid command 35
transaction-logs logging enable command 50
transaction-logs logging entry-type command 50
transaction-logs logging facility command 49
transaction-logs logging host command 49
transaction-logs log-windows-domain command 42, 49
transaction-logs sanitize command 18
transaction-logs sanitized command 43
transaction monitoring
disabling export feature 47
transparent caching
IP spoofing and 10
outgoing proxy exclusion and 5
transparent caching services
disabling 14
enabling 11
transparent forward proxy caching
overview of 6
WCCP-enabled router and 9
transparently received proxy-style requests 6
transparent mode
advanced caching features 14
hierarchical caching in 15
supported caching services in 14
transparent modelist of
supported services 14
transparent redirection
streaming services with 14
supported caching services 14
supported services for 14
troubleshooting
in real time 9
URL performance 5
with debug tool 4
with HTTP CLI client 7
with HTTP statistics tools 2
with log viewing tool 9
with ping 10
with Telnet client 10
with traceroute 10
with WMT diagnostic tools 11
TTL
maximum TTL setting 10
minimum TTL setting 10
Type of Service. See ToS
type-of-service enable command 3
type-tail working.log follow command 9
U
UDI compliance 2
UDP
purpose of 3
unusable multicast address assignments 12
updating the calendar 4
url-filter http bad-sites-deny enable command 14
url-filter http bad-sites-deny file command 14
url-filter http custom-message command 16
url-filter http good-sites-allow enable command 16, 14
url-filter http good-sites-allow file command 16, 14
url-filter http N2H2 allowmode enable command 20
url-filter http smartfilter enable command 39
url-filter http websense allowmode enable command 36
url-filter http websense server command 36
URL filtering
disabling bypass options 12
RADIUS 12
with N2H2 server 12
url-filter rtsp bad-sites-deny file command 15
url-filter rtsp command 15
url-filter rtsp good-sites-allow enable command 15
url-filter rtsp good-sites-allow file command 15

url-filter wmt bad-sites-deny file command 15
url-filter wmt command 15
url-filter wmt good-sites-allow enable command 15
url-filter wmt good-sites-allow file command 15
URLs
  monitoring 5
user accounts
  default account 3
  managing 19
username command 8
UTC
  clock EXEC command 4

V
variable bit rates 24

W
WCCP
  accelerated 34
  access lists 10
  advanced caching features 14
  clean shutdown 18
  configuring
    home router 41
    user-defined services 13
  defining router lists 12
  displaying
    configured services 41
    list of recognized routers 17
  enabling
    on a Content Engine 2
    on routers 23
  enabling on Content Engine 12
  flow protection 9
  flow redirection 10
  information, displaying 42
  IP spoofing 10
  overview of 2
  packet flow diagram 2
  redirected HTTP traffic, accepting 20
  shutting down 17
  transparent forward proxy caching with 9
  Version 1
    enabling on a Content Engine 12
  Version 2
    enabling on a router 20
    multirouter support 8
wccp 14
wccp access-list command 7, 20
WCCP access lists
  configuring 7, 20
  displaying 7, 20
WCCP clustering
  setting parameters for 2
wccp command 10
WCCP-enabled routers
  forward proxy caching and 9
  reverse proxy caching and 10
WCCP flow protection
  configuring 9
  description of 2
  displaying summary information about 9
wccp flow-redirect command
  for WCCP flow protection 9
WCCP flow redirection
  enabling and disabling 17
wccp home-router command 41
wccp https-cache accept-all command 11, 28
wccp https-cache mask command 36, 37
wccp masks
  displaying configurations for 37
wccp port-list command 12
wccp router-list command 13
WCCP service groups
  controlling membership to 23
WCCP services
configuring
  hashing methods 35, 36
masks for 35, 36
port lists for 12
router lists for 13
user-defined services 15
displaying list of 10
dynamic WCCP services 14
list of 3
predefined WCCP services 14
service names 4
service numbers 4
user-defined 14
wccp shutdown max-wait command 18
WCCP slow start
description of 2
wccp slow-start enable command
  WCCP slow start 10
wccp spoof-client-ip enable command 11
WCCP transparent redirection
  overview of 2
WCCP Version 1
  overview of 2
WCCP Version 1 commands
debug ip wccp packets 42
show ip interface 39, 41
show ip wccp web-caches 42
show wccp 42
WCCP Version 2
  Cisco Express Forwarding (CEF) 7
  overview of 2
WCCP Version 2 command
clear ip wccp 34
show wccp 17
wccp web-cache router-list command 14
web cache packet return 6
web objects
description of 2
Websense server
  CLI commands 26
  failover 22
  port configuration 28
  websense-server enable command 35
  websense-server ip-address command 13
  websense-server service edir-agent edir-server administrative-passwd command 26
  websense-server service eim activate command 26
  websense-server service network-agent activate command 26
  websense-server service policy local activate command 26
  websense-server service policy remote command 26
  websense-server service user activate command 26
  websense-server user-server external command 26
Websense services
  activating 34
  deactivating 37
  supported services 24
web server acceleration 10
wildcard character * 7
WMT
caching overview 2
components of 2
configuring
  fast cache 29
  fast start 28
description of 9
displaying request statistics 47
error logging 11
Fast Cache
  configuring on Content Engines 31
Fast Start
  configuring on Content Engines 29
multicasting
  multicast-in and multicast-out 43
  multicast-in and unicast-out 43
  unicast-in and multicast-out 43
  multicast logging 48
removing license key 11
transaction logging 10, 48
unicasting
  unicast-in and unicast-out 44
wmt accelerate command 21
wmt accelerate live-split enable command 21
wmt accelerate proxy-cache enable command 21
wmt accelerate VOD enable command 21
wmt bandwidth incoming bypass-list command 26
wmt broadcast command 43
wmt cache enable command 35
wmt cache max-obj-size command 19, 34, 35
WMT caching proxy
  live splitting 10
  proxy authentication 10
  variable bit rate support 10
wmt disallowed-client-protocols command 18
WMT error logging 55
wmt extended transaction-log enable command 54
wmt fast-cache enable command
  Fast Cache feature
    reenabling 31
wmt fast-proxy-cache command 29
wmt fast-start command 28
wmt http allow extension command 20
WMT license key
  removing 31, 13
WMT live splitting
  accelerating delivery of 21
wmt max-bandwidth command 19
wmt max-bitrate command 19
WMT multicast-in multicast out
  configuring 38
  description of 38
WMT multicast-in unicast out
  configuring 43
  description of 43
wmt multicast log command 48
WMT multicasts
  source failover for 42
  sources of 41
WMT multicasts, types of
  multicast-in multicast-out 38, 39
  multicast-in unicast-out 43
  unicast-in unicast-out 44
wmt multicast schedule-start command 41
wmt multicast-station command 42
wmt multicast station-configuration station-name failover command 42
WMT multicast stations
  configuring schedules for 43
  starting and stopping 41
wmt multicast time-to-live command
  WMT multicasts
    TTL for 38
WMT proxy caching
  accelerating delivery of 21
  configuring
    through Content Engine CLI 15
    through Setup utility 15
wmt proxy outgoing http host command 11, 19
WMT streaming and caching
  displaying WMT statistics 47
WMT streaming and caching services
  description of 5
  list of 5
WMT streams
  accelerating delivery of 21
  clearing on Content Engine 45
WMT transaction log
  logging usernames 54
  specifying format of 50
wmt transaction-logs format command 50
WMT transparent caching
  configuring
    through CLI 15
    through Setup utility 15
WMT unicast-in multicast out
configuring 39
description of 39
WMT unicast-in unicast out
configuring 44
description of 44
WMT VOD files
accelerating delivery of 21
write memory command 13