tcp

To configure Transmission Control Protocol (TCP) parameters, use the **tcp** global configuration command. To disable TCP parameters, use the **no** form of this command.

tcp client-mss *maxsegsize*

tcp client-receive-buffer *kbytes*

tcp client-rw-timeout *seconds*

tcp client-satellite

tcp client-send-buffer *kbytes*

tcp cwnd-base *segments*

tcp ecn enable

tcp increase-xmit-timer-value *value*

tcp init-ss-threshold *value*

tcp keepalive-probe-cnt *count*

tcp keepalive-probe-interval *seconds*

tcp keepalive-timeout *seconds*

tcp memory-limit {**low-water-mark** *megabytes* | **high-water-mark-pressure** *megabytes* | **high-water-mark-absolute** *megabytes*}

tcp server-mss *maxsegsize*

tcp server-receive-buffer *kbytes*

tcp server-rw-timeout *seconds*

tcp server-satellite

tcp server-send-buffer *kbytes*

tcp type-of-service enable

no tcp {**client-mss** | **client-receive-buffer** | **client-rw-timeout** | **client-satellite** | **client-send-buffer** | **cwnd-base** | **ecn** | **increase-xmit-timer-value** | **init-ss-threshold** | **keepalive-probe-cnt** | **keepalive-probe-interval** | **keepalive-timeout** | **memory-limit** | **server-mss** | **server-receive-buffer** | **server-rw-timeout** | **server-satellite** | **server-send-buffer** | **type-of-service**}

Syntax Description

client-mss	Sets the TCP maximum segment size sent to clients.
maxsegsize	Maximum segment size in bytes (512–1460).

client-receive-buffer	Sets the client connection's receive buffer size (the TCP incoming window size).
kbytes	Receive buffer size in kilobytes (1–512).
client-rw-timeout	Sets the interval after which the Content Engine times out trying to read or write to the network.
seconds	Timeout in seconds (1–3600).
client-satellite	Sets the client TCP compliance to RFC 1323 standard.
client-send-buffer	Sets client connection's send buffer size (the TCP outgoing window size).
kbytes	Send the buffer size in kilobytes (8–512).
cwnd-base	Sets the initial send congestion window in segments.
segments	Initial send congestion window segments (1–10).
ecn enable	Enables the TCP explicit congestion notification.
increase-xmit-timer-value	Factor (1–3) used to modify the length of the retransmit timer by 1 to 3 times the base value determined by the TCP algorithm. **Note** Modify this factor with caution. It can improve throughput when TCP is used over slow reliable connections but should never be changed in an unreliable packet delivery environment.
value	Retransmit multiple (1–3).
init-ss-threshold	Sets the initial slow-start threshold value.
value	Slow-start threshold value.
keepalive-probe-cnt	Sets the length of time that the Content Engine keeps an idle connection open.
count	Number of probe counts (1–10).
keepalive-probe-interval	Sets the number of times that the Content Engine retries a connection.
seconds	Keepalive probe interval in seconds (1–300).
keepalive-timeout	Sets the length of time that the Content Engine keeps a connection open before disconnecting.
seconds	Keepalive timeout in seconds (1–3600).
memory-limit	Specifies the limits for system memory usage by TCP. These limits include send and receive buffer usage of all connections.
low-water-mark	Specifies the lower limit (in megabytes) of memory pressure mode, below which TCP enters into normal memory allocation mode. The range is 4–600.
high-water-mark-pressure	Specifies the upper limit (in megabytes) of normal memory allocation mode, beyond which TCP enters into memory pressure mode. The range is 5–610.
high-water-mark-absolute	Specifies the absolute limit (in megabytes) on TCP memory usage. The range is 6–620.
server-mss	Sets the TCP maximum segment size sent to servers.
maxsegsize	Maximum segment size in bytes (512–1460).
server-receive-buffer	Sets the server connection's receive buffer size (the TCP incoming window size).
kbytes	Receive buffer size in kilobytes (1–512).

server-rw-timeout	Sets the interval after which the Content Engine times out trying to read or write to the network.
seconds	Read/write timeout in seconds (1–3600).
server-satellite	Sets the server TCP compliance to the RFC 1323 standard.
server-send-buffer	Sets the server connection's send buffer size (the TCP outgoing window size).
kbytes	Send buffer size in kilobytes (8–512).
type-of-service enable	Sets the TCP Type of Service to match the client's Type of Service.

Defaults

tcp server maximum segment size: 1460 bytes

tcp client maximum segment size: 1432 bytes

tcp server-receive-buffer: 32 KB

tcp client-receive-buffer: 8 KB

tcp server-rw-timeout: 120 seconds

tcp client-rw-timeout: 120 seconds

tcp server-send-buffer: 8 KB

tcp client-send-buffer: 32 KB

tcp keepalive-probe-cnt: 4

tcp keepalive-probe-interval: 75 seconds

tcp keepalive-timeout: 300 seconds

tcp server-satellite (RFC 1323): disabled

tcp client-satellite (RFC 1323): disabled

tcp type of service: disabled

tcp cwnd-base: 2 segments

tcp increase-xmit-timer-value: 1

tcp init-ss-threshold: 2 segments

Table 2-172 gives the default TCP memory limit settings.

Table 2-172 Default TCP Memory Limit Settings

Total System Memory	Low	Pressure	Absolute
1 GB, 2 GB, or 4 GB	360 MB	380 MB	400 MB
512 MB	180 MB	190 MB	200 MB
256 MB	25 MB	28MB	30 MB

Command Modes

global configuration

Usage Guidelines

Caches are typically deployed 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

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.

Caution Because of the complexities involved in TCP parameters, be careful when tuning these parameters. In nearly all environments, the default TCP settings are adequate. Do not fine tune the TCP settings unless you are an experienced network administrator who fully understands the TCP operation details.

Explicit Congestion Notification

TCP Explicit Congestion Notification (ECN) allows an intermediate router to notify the end hosts of impending network congestion. It also provides enhanced support for TCP sessions associated with applications that are sensitive to delay or packet loss, including Telnet, web browsing, and transfer of audio and video data. The major issue with ECN is the need to change the operation of both the routers and the TCP software stacks to accommodate the operation of ECN.

Congestion Windows

The congestion window (*cwnd*) is a TCP state variable that limits the amount of data that a TCP sender can transmit onto the network before receiving an acknowledgment (ACK) from the receiving side of the TCP transmission. The TCP *cwnd* variable is implemented by the TCP congestion avoidance algorithm. The goal of the congestion avoidance algorithm is to continually modify the sending rate so that the sender automatically senses any increase or decrease in available network capacity during the entire data flow. When congestion occurs (manifested as a packet loss), the sending rate is first lowered and then gradually increased as the sender continues to probe the network for additional capacity.

Retransmit Time Multiplier

The TCP sender uses a timer to measure the time that has elapsed between sending a data segment and receiving the corresponding ACK from the receiving side of the TCP transmission. When this retransmit timer expires, the sender (according to the RFC standards for TCP congestion control) must reduce its sending rate. However, because the sender is not reducing its sending rate in response to network congestion, the sender is not able to make any valid assumptions about the current state of the network. To avoid congesting the network with an inappropriately large burst of data, the sender implements the slow-start algorithm, which reduces the sending rate to one segment per transmission.

You can modify the sender's retransmit timer by using the **tcp increase-xmit-timer-value** global configuration command. The retransmit time multiplier modifies the length of the retransmit timer by one to three times the base value, as determined by the TCP algorithm that is being used for congestion control.

When making adjustments to the retransmit timer, be aware that they affect performance and efficiency. If the retransmit timer is triggered too early, the sender pushes duplicate data onto the network unnecessarily; if the timer is triggered too slowly, the sender remains idle for too long, unnecessarily slowing the data flow.

TCP Slow Start

Slow start is one of four congestion control algorithms used by TCP. The slow-start algorithm controls the amount of data being inserted into the network at the beginning of a TCP session when the capacity of the network is not known.

For example, if a TCP session began by inserting a large amount of data into the network, much of the initial burst of data would probably be lost. Instead, TCP initially transmits a modest amount of data that has a high probability of successful transmission. TCP then probes the network by sending increasing amounts of data as long as the network does not show signs of congestion.

The slow-start algorithm begins by sending packets at a rate that is determined by the congestion window or *cwnd* variable. The algorithm continues to increase the sending rate until it reaches the limit set by the slow-start threshold (*ssthresh*) variable. (Initially, the value of the *ssthresh* variable is adjusted to the receiver's maximum window size [RMSS]. However, when congestion occurs, the *ssthresh* variable is set to half the current value of the *cwnd* variable, marking the point of the onset of network congestion for future reference.)

The starting value of the *cwnd* variable is set to that of the sender maximum segment size (SMSS), which is the size of the largest segment that the sender can transmit. The sender sends a single data segment. Because the congestion window is equal to the size of one segment, the congestion window is now full. The sender then waits for the corresponding ACK from the receiving side of the transmission. When the ACK is received, the sender increases its congestion window size by increasing the value of the *cwnd* variable by the value of one SMSS. Now the sender can transmit two segments before the congestion window is again full and the sender is once more required to wait for the corresponding ACKs for these segments. The slow-start algorithm continues to increase the value of the *cwnd* variable and increase the size of the congestion window by one SMSS for every ACK received. If the value of the *cwnd* variable increases beyond the value of the *ssthresh* variable, then the TCP flow control algorithm changes from the slow-start algorithm to the congestion avoidance algorithm.

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 the following 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. The 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.

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

TCP Read-Write Timeout and 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 keeps 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 send 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 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.

Configuring Content Engines to Send out TCP Keepalives

By default, the Content Engine does not automatically send out keepalives. To configure a 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 sends 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 is not received, the Content Engine waits 90 seconds and logs a miss. After four misses, the Content Engine considers the HTTP connection is down and closes the connection.

Use the **tcp keepalive-probe-cnt** global configuration command to specify how many times the Content Engine should attempt to connect to the device before closing the connection. The count can be from 1 to 10; the default is 4 attempts.

Use the **tcp keepalive-probe-interval** global configuration command to specify how often the Content Engine is to send out a TCP keepalive. The interval can be from 1 to 120 seconds; the default is 75 seconds.

Use the **tcp keepalive-timeout** global configuration command to wait for a response (the device does not respond) before the Content Engine logs a miss. The timeout can be from 1 to 120 seconds; the default is 90 seconds.

Configuring TCP Memory Limits

In the ACNS 5.3.3 software and later releases, you can configure TCP memory limits. 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. Use the **tcp memory-limit** global configuration command to configure TCP memory limits.

Caution Do not modify the default values unless you are an experienced network administrator who fully understands the TCP memory limits. The default values are device dependent and have been chosen after extensive testing. The default values should not be changed under normal conditions. Increasing these values can cause the TCP subsystem to use more memory, which might render the system unresponsive. Decreasing these values can cause increased response times and lower the network's performance.

Examples

The following example shows how to configure the TCP receiving buffer size in kilobytes for incoming TCP packets:

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

The following example shows how to reset the TCP receiving buffer size in kilobytes for incoming TCP packets:

```
ContentEngine(config)# no tcp client-receive-buffer
```

Note The client and server receive and send buffer settings have only an indirect impact on the advertised TCP window size. The buffer setting is the approximate and not the exact limit that is used by the system.

Related Commands

clear statistics tcp
show statistics tcp
show tcp

tcpdump

To dump the network traffic, use the **tcpdump** EXEC command.

tcpdump [*LINE*]

Syntax Description

LINE	(Optional) Specifies the dump options.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use the **tcpdump** command to gather a sniffer trace on the Content Engine, Content Router, or Content Distribution Manager for troubleshooting, when asked to gather the data by the Cisco TAC. See the "Obtaining Technical Assistance" section on page xvii for more information. This utility is very similar to the Linux or Unix **tcpdump** command.

The **tcpdump** command allows an administrator (must be an admin user) to capture packets from the Ethernet. On the Content Engine 500 series, the interface names are eth0 and eth1. On all ACNS platforms, we recommend that you specify a path/filename in the local1 directory.

You can do a straight packet header dump to the screen by entering the **tcpdump** command. Press **Ctrl-C** to stop the dump.

The **tcpdump** command has the following options:

- **-w** <*filename*>—writes the raw packet capture output to a file.
- **-s** <*count*>—captures the first <count> bytes of each packet.
- **-i** <*interface*>—allows you to specify a specific interface to use for capturing the packets.
- **-c** <*count*>—limits the capture to <count> packets.

The following example captures the first 1500 bytes of the next 10,000 packets from interface Ethernet 0 and puts the output in a file named dump.pcap in the local1 directory on the Content Engine:

```
ContentEngine# tcpdump -w /local1/dump.pcap -i eth0 -s 1500 -c 10000
```

When you specify the **-s** option, it sets the packet snaplength. The default value captures only 64 bytes, and this default setting saves only packet headers into the capture file. For troubleshooting of redirected packets or higher level traffic (HTTP, authentication, and so on), you must copy the complete packets.

After the TCP dump has been collected, you need to move the file from the Content Engine to a PC so that the file can be viewed by a sniffer decoder.

```
ftp <ip address of the CE>

!--- Log in using the admin username and password.


    cd local1
    bin
```

```
     hash
     get <name of the file>


!--- Using the above example, it would be dump.pcap.

     bye
```

We recommend that you use Ethereal as the software application for reading the TCP dump. With Ethereal, you can decode packets that are encapsulated into a GRE tunnel used by WCCP redirection. See the Ethereal website for further information.

Note In most cases, redirected packets captured by the tcpdump facility with the ACNS CLI differ from the data received on the interface. The destination IP address and TCP port number are modified to reflect the device IP address and the port number 8999.

Examples

The following example shows how to dump the TCP network traffic:

```
CONTENTENGINE# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 68 bytes
12:45:42.617677 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P
3342832089:3342832201(112) ack 1248615673 win 15232
12:45:42.618950 IP 172.19.226.63 > CONTENTENGINE.cisco.com: icmp 36: 172.19.226.63 udp
port 2048 unreachable
12:45:42.619327 IP CONTENTENGINE.cisco.com.10015 > dns-sj2.cisco.com.domain:
49828+[|domain]
12:45:42.621158 IP dns-sj2.cisco.com.domain > CONTENTENGINE.cisco.com.10015:  49828
NXDomain*[|domain]
12:45:42.621942 IP CONTENTENGINE.cisco.com.10015 > dns-sj2.cisco.com.domain:
49829+[|domain]
12:45:42.623799 IP dns-sj2.cisco.com.domain > CONTENTENGINE.cisco.com.10015:  49829
NXDomain*[|domain]
12:45:42.624240 IP CONTENTENGINE.cisco.com.10015 > dns-sj2.cisco.com.domain:
49830+[|domain]
12:45:42.626164 IP dns-sj2.cisco.com.domain > CONTENTENGINE.cisco.com.10015:
49830*[|domain]
12:45:42.702891 802.1d config TOP_CHANGE 8000.00:03:9f:f1:10:63.8042 root
8000.00:01:43:9a:c8:63 pathcost 26 age 3 max 20 hello 2 fdelay 15
12:45:42.831404 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 112 win 64351
12:45:42.831490 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: . 112:1444(1332) ack 1
win 15232
12:45:42.831504 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 1444:1568(124) ack 1
win 15232
12:45:42.831741 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 1568:1696(128) ack 1
win 15232
12:45:43.046176 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 1568 win 65535
12:45:43.046248 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 1696:2128(432) ack 1
win 15232
12:45:43.046469 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 2128:2256(128) ack 1
win 15232
12:45:43.046616 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 2256:2400(144) ack 1
win 15232
12:45:43.107700 802.1d config TOP_CHANGE 8000.00:03:9f:f1:10:63.8042 root
8000.00:01:43:9a:c8:63 pathcost 26 age 3 max 20 hello 2 fdelay 15
12:45:43.199710 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 1696 win 65407
12:45:43.199784 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 2400:2864(464) ack 1
win 15232
```

```
12:45:43.199998 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 2864:2992(128) ack 1
win 15232
12:45:43.259968 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 2400 win 64703
12:45:43.260064 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 2992:3280(288) ack 1
win 15232
12:45:43.260335 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 3280:3408(128) ack 1
win 15232
12:45:43.260482 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 3408:3552(144) ack 1
win 15232
12:45:43.260621 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 3552:3696(144) ack 1
win 15232
12:45:43.413320 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 2992 win 65535
12:45:43.413389 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 3696:3984(288) ack 1
win 15232
12:45:43.413597 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 3984:4112(128) ack 1
win 15232
12:45:43.413741 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 4112:4256(144) ack 1
win 15232
12:45:43.473601 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 3552 win 64975
12:45:43.473659 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 4256:4544(288) ack 1
win 15232
12:45:43.473853 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 4544:4672(128) ack 1
win 15232
12:45:43.473994 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 4672:4816(144) ack 1
win 15232
12:45:43.474132 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 4816:4960(144) ack 1
win 15232
12:45:43.484117 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: P 1:81(80) ack 3696
win 64831
12:45:43.484167 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 4960:5248(288) ack
81 win 15232
12:45:43.484424 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 5248:5392(144) ack
81 win 15232
12:45:43.627125 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 4112 win 64415
12:45:43.627204 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 5392:5680(288) ack
81 win 15232
12:45:43.627439 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 5680:5808(128) ack
81 win 15232
12:45:43.627586 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 5808:5952(144) ack
81 win 15232
12:45:43.688261 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 4544 win 65535
12:45:43.688316 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 5952:6240(288) ack
81 win 15232
12:45:43.688495 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 6240:6368(128) ack
81 win 15232
12:45:43.688638 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 6368:6512(144) ack
81 win 15232
12:45:43.689012 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 4960 win 65119
12:45:43.689046 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 6512:6800(288) ack
81 win 15232
12:45:43.689170 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 6800:6928(128) ack
81 win 15232
12:45:43.689309 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 6928:7072(144) ack
81 win 15232
12:45:43.689447 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 7072:7216(144) ack
81 win 15232
12:45:43.698391 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 5392 win 64687
12:45:43.698437 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 7216:7504(288) ack
81 win 15232
12:45:43.698599 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 7504:7632(128) ack
81 win 15232
12:45:43.698740 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 7632:7776(144) ack
81 win 15232
12:45:43.840558 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 5808 win 64271
```

```
12:45:43.840622 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 7776:8064(288) ack
81 win 15232
12:45:43.840819 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 8064:8192(128) ack
81 win 15232
12:45:43.840962 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 8192:8336(144) ack
81 win 15232
12:45:43.901868 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 6368 win 65535
12:45:43.901938 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 8336:8624(288) ack
81 win 15232
12:45:43.901887 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 6928 win 64975
12:45:43.901910 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 7216 win 64687
12:45:43.902137 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 8624:8752(128) ack
81 win 15232
12:45:43.902281 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 8752:8896(144) ack
81 win 15232
12:45:43.902414 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 8896:9024(128) ack
81 win 15232
12:45:43.902547 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 9024:9152(128) ack
81 win 15232
12:45:43.902687 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 9152:9296(144) ack
81 win 15232
12:45:43.902826 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 9296:9440(144) ack
81 win 15232
12:45:43.902965 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 9440:9584(144) ack
81 win 15232
12:45:43.903104 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 9584:9728(144) ack
81 win 15232
12:45:43.922413 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 7632 win 64271
12:45:43.922459 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 9728:10304(576) ack
81 win 15232
12:45:43.922622 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 10304:10432(128) ack
81 win 15232
12:45:43.922764 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 10432:10576(144) ack
81 win 15232
12:45:44.053872 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 8192 win 65535
12:45:44.053972 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 10576:10864(288) ack
81 win 15232
12:45:44.054308 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 10864:11104(240) ack
81 win 15232
12:45:44.054453 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 11104:11248(144) ack
81 win 15232
12:45:44.054596 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 11248:11392(144) ack
81 win 15232
12:45:44.111702 802.1d config TOP_CHANGE 8000.00:03:9f:f1:10:63.8042 root
8000.00:01:43:9a:c8:63 pathcost 26 age 3 max 20 hello 2 fdelay 15
12:45:44.114626 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 8752 win 64975
12:45:44.114712 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 11392:11712(320) ack
81 win 15232
12:45:44.115219 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 11712:11952(240) ack
81 win 15232
12:45:44.115381 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 11952:12096(144) ack
81 win 15232
12:45:44.115426 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 9152 win 64575
12:45:44.115617 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 12096:12336(240) ack
81 win 15232
12:45:44.115760 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 12336:12480(144) ack
81 win 15232
12:45:44.115904 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 12480:12624(144) ack
81 win 15232
12:45:44.116045 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 12624:12768(144) ack
81 win 15232
12:45:44.116094 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 9440 win 64287
12:45:44.116114 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 9728 win 65535
```

```
12:45:44.116332 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 12768:13088(320) ack
81 win 15232
12:45:44.116473 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 13088:13232(144) ack
81 win 15232
12:45:44.116614 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 13232:13376(144) ack
81 win 15232
12:45:44.116755 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 13376:13520(144) ack
81 win 15232
12:45:44.116895 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 13520:13664(144) ack
81 win 15232
12:45:44.135947 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: . ack 10432 win 64831
12:45:44.135996 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 13664:13808(144) ack
81 win 15232
12:45:44.136223 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 13808:14048(240) ack
81 win 15232
12:45:44.136366 IP CONTENTENGINE.cisco.com.ssh > 10.77.140.97.4314: P 14048:14192(144) ack
81 win 15232
12:45:44.144104 IP 10.77.140.97.4314 > CONTENTENGINE.cisco.com.ssh: P 81:161(80) ack 10576
win 64687

102 packets captured
105 packets received by filter
0 packets dropped by kernel
```

The following example shows how to dump the TCP network traffic and redirect it to a file named test:

```
CONTENTENGINE# tcpdump port 8080  -w test
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 68 bytes
216 packets captured
216 packets received by filter
0 packets dropped by kernel
```

telnet

To log in to a network device using the Telnet client, use the **telnet** EXEC command.

telnet {*hostname* | *ip-address*} [*portnum*]

Syntax Description

hostname	Hostname of the network device.
ip-address	IP address of the network device.
portnum	(Optional) Port number (1–65535). Default port number is 23.

Defaults

The default port number is 23.

Command Modes

EXEC

Usage Guidelines

Some UNIX shell functions, such as escape and the **suspend** command, are not available in the Telnet client. In addition, multiple Telnet sessions are also not supported.

This Telnet client allows you to specify a destination port. By entering this command, you can test websites by attempting to open a Telnet session to the website from the Content Engine CLI.

Examples

The following example shows how to open a Telnet session to a network device using the hostname:

```
ContentEngine# telnet cisco-ce
```

The following example shows how to open a Telnet session to a network device using the IP address:

```
ContentEngine# telnet 172.16.155.224
```

The following example shows how to open a Telnet session to a network device on port 8443 using the hostname:

```
ContentEngine# telnet cisco-ce 8443
```

The following example shows how to open a Telnet session to a network device on port 80 using the hostname:

```
ContentEngine# telnet www.yahoo.com 80
```

telnet enable

To enable Telnet, use the **telnet enable** global configuration command. To disable Telnet, use the **no** form of this command.

telnet enable

no telnet enable

Syntax Description

This command has no arguments or keywords.

Defaults

Enabled

Command Modes

global configuration

Usage Guidelines

Use this terminal emulation protocol for a remote terminal connection. The **telnet enable** command allows users to log in to other devices using a Telnet session.

Examples

The following example shows how to enable Telnet on the Content Engine:

```
ContentEngine(config)# telnet enable
```

Related Commands

show telnet

terminal

To set the number of lines displayed in the console window, or to display the current console **debug** command output, use the **terminal** EXEC command.

terminal {**length** *length* | **monitor** [**disable**]}

Syntax Description

length	Sets the length of the display on the terminal.
length	Length of the display on the terminal (0–512). Setting the length to 0 means that there is no pausing.
monitor	Copies the debug output to the current terminal.
disable	(Optional) Disables monitoring at this specified terminal.

Defaults

The default is 24 lines.

Command Modes

EXEC

Usage Guidelines

When 0 is entered as the *length* parameter, the output to the screen does not pause. For all nonzero values of *length*, the -More- prompt is displayed when the number of output lines matches the specified *length* number. The -More- prompt is considered a line of output. To view the next screen, press the **Spacebar**. To view one line at a time, press the **Enter** key.

The **terminal monitor** command allows a Telnet session to display the output of the **debug** commands that appear on the console. Monitoring continues until the Telnet session is terminated.

Examples

The following example sets the number of lines to display to 20:

```
ContentEngine# terminal length 20
```

The following example configures the terminal for no pausing:

```
ContentEngine# terminal length 0
```

Related Commands

All **show** commands

test-url

To test the accessibility of a URL, using FTP, HTTP, or HTTPS, use the **test-url** EXEC command.

test-url {**ftp** *url* [**use-ftp-proxy** *proxy-url*] | **http** *url* [**custom-header** *header* [**head-only**] [**use-http-proxy** *proxy-url*] | **head-only** [**custom-header** *header*] [**use-http-proxy** *proxy-url*] | **use-http-proxy** *proxy-url* [**custom-header** *header*] [**head-only**]] | **https** *url* [**head-only**]}

Syntax Description

ftp	Specifies the FTP URL to be tested.
url	FTP URL to be tested. Use one of the following formats to specify the FTP URL: • ftp://*domainname*/*path* • ftp://*user*:*password*@*domainname*/*path*
use-ftp-proxy	(Optional) Specifies the FTP proxy that is used to test the URL.
proxy-url	FTP proxy URL. Use one of the following formats to specify the proxy URL: • *proxy IP Address*:*proxy Port* • *proxy Username*:*proxy Password*@*proxy IP Address*:*proxy Port*
http	Specifies the HTTP URL to be tested.
url	HTTP URL to be tested. Use one of the following formats to specify the HTTP URL: • http://*domainname*/*path* • http://*user*:*password*@*domainname*/*path*
custom-header	(Optional) Specifies the custom header information to be sent to the server.
header	Custom header information to be sent to the server. Use the format *header*:*line* to specify the custom header.
head-only	(Optional) Specifies that only the HTTP header information must be retrieved.
use-http-proxy	(Optional) Specifies the HTTP proxy that is used to test the URL.
proxy-url	HTTP proxy URL. Use one of the following formats to specify the HTTP proxy URL: http://*proxyIp*:*proxyPort* http://*proxyUser*:*proxypasswd*@*proxyIp*:*proxyPort*
https	Specifies the HTTPS URL to be tested.
url	HTTPS URL to be tested. Use one of the following formats to specify the HTTPS URL: • https://*domainname*/*path* • https://*user*:*password*@*domainname*/*path*
head-only	(Optional) Specifies that only the HTTPS header information must be retrieved.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

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 in the ACNS 5.2 software and later releases allows the users to test whether a URL is accessible over the FTP, HTTP, and HTTPS protocols. When you test the connectivity using the **test-url** command, the Content Engine sends a request using the protocol that you have specified to the server and fetches the requested contents. The actual content is dumped into the path /dev/null, and the server response with the header information is displayed to the user.

You can use the **test-url ftp** command to test the following for the specified URL:

- Connectivity to the URL
- Connectivity to the URL through the FTP proxy (using the **use-ftp-proxy** option)
- Authentication
- FTP proxy authentication

You can use the **test-url http** command to test the following for the specified URL:

- Test the connectivity to the URL
- Test the connectivity to the URL through the HTTP proxy (using the **use-http-proxy** option)
- Authentication
- HTTP proxy authentication
- Header information only for the specified page (using the **head-only** option) or additional header information (using the **custom-header** option)

You can use the **test-url https** command to test whether the HTTPS URL is accessible and to test authentication to the web server. You can use the **head-only** option to request only the header information alone for the specified page.

Examples

The following example tests the accessibility to the URL http://192.168.171.22 using HTTP:

```
ContentEngine# test-url http http://ce1.server.com
--02:27:20--  http://ce1.server.com/
           => `/dev/null'
Len - 22 , Restval - 0 , contlen - 0 , Res - 134728056Resolving ce1.server.com..

done.
Connecting to ce1.server.com[192.168.171.22]:80... connected.
HTTP request sent, awaiting response...
 1 HTTP/1.1 200 OK
 2 Date: Mon, 26 Jul 2004 08:41:34 GMT
 3 Server: Apache/1.2b8
 4 Last-Modified: Fri, 25 Apr 2003 12:23:04 GMT
 5 ETag: "1aee29-663-3ea928a8"
 6 Content-Length: 1635
 7 Content-Type: text/html
 8 Via: 1.1 Application and Content Networking System Software 5.2
 9 Connection: Keep-Alive
 (1635 to go)
 0% [                                     ] 0             --.--K/s    ETA --:--L
en - 0   ELen - 1635     Keepalive - 1
100%[====================================>] 1,635          1.56M/s    ETA 00:00

02:27:20 (1.56 MB/s) - `/dev/null' saved [1635/1635]
```

The following example tests the accessibility to the URL http://192.168.171.22 through the HTTP proxy 10.107.192.148:

```
ContentEngine# test-url http http://192.168.171.22 use-http-proxy 10.107.192.148:8090
--15:22:51--  http://10.77.155.246/
           => `/dev/null'
Len - 1393 , Restval - 0 , contlen - 0 , Res - 134728344Connecting to
10.107.192.148:8090... connected.
Proxy request sent, awaiting response...
 1 HTTP/1.1 401 Authorization Required
 2 Date: Mon, 27 Sep 2004 15:29:18 GMT
 3 Server: Apache/1.3.27 (Unix) tomcat/1.0
 4 WWW-Authenticate: Basic realm="IP/TV Restricted Zone"
 5 Content-Type: text/html; charset=iso-8859-1
 6 Via: 1.1 Application and Content Networking System Software 5.2.1
 7 Connection: Close
Len - 0 , Restval - 0 , contlen - -1 , Res - -1Connecting to 10.107.192.148:8090...
connected.
Proxy request sent, awaiting response...
 1 HTTP/1.1 401 Authorization Required
 2 Date: Mon, 27 Sep 2004 15:29:19 GMT
 3 Server: Apache/1.3.27 (Unix) tomcat/1.0
 4 WWW-Authenticate: Basic realm="IP/TV Restricted Zone"
 5 Content-Type: text/html; charset=iso-8859-1
 6 Via: 1.1 Application and Content Networking System Software 5.2.1
 7 Connection: Keep-Alive
 (1635 to go)
 0% [                                    ] 0             --.--K/s    ETA --:--L
en - 0   ELen - 1635     Keepalive - 1
100%[====================================>] 1,635          1.56M/s    ETA 00:00

02:27:20 (1.56 MB/s) - `/dev/null' saved [1635/1635]
```

The following example shows that 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 - 134728056Resolving gmail.google.com
...
Resolving gmail.google.com failed: Host not found.
```

The following example shows that the test URL is an FTP-over-HTTP request using the Content Engine that has an IP address of 10.77.157.42 as the FTP proxy for the request:

```
ContentEngine# test-url ftp ftp://1234567:pAB6rB7x@smartfilter.com  use-ftp-proxy ?
     WORD  Proxy-url. Format http://proxyIp:proxyPort or
     http://proxyUser:proxypasswd@proxyIp:proxyPort
ContentEngine# test-url ftp ftp://ssivakum:ssivakum@10.77.157.148/antinat-0.90.tar
use-ftp-proxy 10.77.157.42:8080
Mar 30 14:34:40 nramaraj-ce admin-shell: %CE-PARSER-6-350232: CLI_LOG shell_parser_log:
test-url ftp ftp://ssivakum:ssivakum@10.77.157.148/antinat-0.90.tar use-ftp-proxy
10.77.157.42:8080
--14:34:40--  ftp://ssivakum:*password*@10.77.157.148/antinat-0.90.tar
           => `/dev/null'
Len - 1185 , Restval - 0 , contlen - 0 , Res - 134728552Connecting to 10.77.157.42:8080...
connected.
Proxy request sent, awaiting response...
 1 HTTP/1.0 200 OK
 2 Mime-Version: 1.0
 3 Server: Application and Content Networking System Software 5.1.14
 4 Content-Type: application/x-tar
 5 Content-Length: 1771520
 6 Last-Modified: Wed Mar 30 03:53:05 2005
```

```
 7 X-protocol: ftp
 8 Date: Wed, 30 Mar 2005 14:04:27 GMT
 9 Connection: Close
 (1771520 to go)
 0% [
] 0             --.--K/s    ETA --:--Len - 1460        ELen - 1771520          Keepalive -
0
100%[=======================================================================================
>] 1,771,520    145.01K/s    ETA 00:00

14:34:53 (145.01 KB/s) - `/dev/null' saved [1771520/1771520]

14:34:53 URL:ftp://10.77.157.148/antinat-0.90.tar [1771520/1771520]
ContentEngine#
```

The following example tests the accessibility to the URL ftp://ssivakum:ssivakum@10.77.157.148 using FTP:

```
ContentEngine# test-url ftp ftp://ssivakum:ssivakum@10.77.157.148/antinat-0.90.tar
Mar 30 14:33:44 nramaraj-ce admin-shell: %CE-PARSER-6-350232: CLI_LOG shell_parser_log:
test-url ftp ftp://ssivakum:ssivakum@10.77.157.148/antinat-0.90.tar
--14:33:44--  ftp://ssivakum:*password*@10.77.157.148/antinat-0.90.tar
           => `/dev/null'
Connecting to 10.77.157.148:21... connected.
Logging in as ssivakum ...
220 (vsFTPd 1.1.3)
--> USER ssivakum

331 Please specify the password.
--> PASS Turtle Power!
230 Login successful. Have fun.
--> SYST

215 UNIX Type: L8
--> PWD

257 "/home/ssivakum"
--> TYPE I

200 Switching to Binary mode.
==> CWD not needed.
--> PORT 10,1,1,52,82,16

200 PORT command successful. Consider using PASV.
--> RETR antinat-0.90.tar

150 Opening BINARY mode data connection for antinat-0.90.tar (1771520 bytes).
Length: 1,771,520 (unauthoritative)

 0% [
] 0             --.--K/s    ETA --:--Len - 0   ELen - 1771520          Keepalive - 0
100%[=======================================================================================
>] 1,771,520    241.22K/s    ETA 00:00

226 File send OK.
14:33:53 (241.22 KB/s) - `/dev/null' saved [1771520]

ContentEngine#
```

Related Commands **acquirer test-url**

tftp-server

To configure the Content Engine to act as a Trivial File Transfer Protocol (TFTP) server or as a gateway that serves TFTP requests by accessing content on HTTP or FTP servers, use the **tftp-server** global configuration command. To disable TFTP server options, use the **no** form of this command.

tftp-server access-list {*std-acl-name* | *std-acl-num*}

tftp-server dir *directory*

tftp-server gw proto {**http** | **ftp**} **server** {*hostname* | *ip_address*} **pri** *priority* [**name** *name* **passwd** *password*] [**path** *directory*]

no tftp-server {**access-list** [*std-acl-name* | *std-acl-num*] | **dir** *directory* | **gw proto** {**http** | **ftp**} **server** {*hostname* | *ip_address*} **pri** *priority* [**name** *name* **passwd** *password*] [**path** *directory*]}

Syntax Description

access-list	Identifies the standard access list that permits or denies access to the TFTP service running on the Content Engine.
std-acl-name	Standard access list name. Alphanumeric identifier of up to 30 characters, beginning with a letter that identifies the ACL to apply to the current interface.
std-acl-num	Standard access list number (1–99). Numeric identifier that identifies the access list to apply to the current interface.
dir	Sets the local Content Engine directory to search for requested files.
directory	Local Content Engine directory to search for requested files.
gw	Configures the gateway functionality for the Content Engine.
proto	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.
http	Uses the HTTP protocol to access the origin server.
ftp	Uses the FTP protocol to access the origin server.
server	Configures the origin server.
hostname	Hostname of the origin server.
ip-address	IP address of the origin server.
pri	Sets the priority of the origin server.
priority	Priority (1 or 2) of the origin server.
name	(Optional) Sets the username for authentication to the origin server.
name	(Optional) Username for authentication to the origin server.
passwd	(Optional) Sets the password for authentication to the origin server.
password	(Optional) Password for authentication to the origin server.
path	(Optional) Sets the pathname to search for files on the origin server.
directory	(Optional) Pathname to search for files on the origin server.

Defaults

By default, 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. The number of retries is fixed at five retries. These values are not configurable.

Maximum size for a transferred file is 32 MB.

Command Modes

global configuration

Usage Guidelines

In the ACNS software, the Content Engine can act as a TFTP server and process TFTP requests sent to the Content Engine. In addition, the Content Engine can act as a TFTP gateway, allowing users to obtain files from HTTP or FTP servers in response to a TFTP request.

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.

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.

Using the TFTP Server or Gateway

You can use the **tftp-server** command to configure the TFTP server and gateway to serve the content in response to TFTP requests in three ways:

- Serving local content (TFTP server)—Available on any ACNS device
- Serving pre-positioned content (TFTP gateway)—Available only on a Content Engine
- Serving content from remote servers (TFTP gateway)—Available only on a Content Engine

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 in a TFTP server directory, the Content Engine forwards the request to the caching application using the HTTP or FTP protocol, and one of the following events occurs:

- If the file is already cached or pre-positioned, the Content Engine sends the file to the client that issued the request.
- If the file is found on a remote server, the Content Engine caches the file and sends it to the client; it serves subsequent requests for the file from its local cache.
- If the file is not found, the Content Engine replies to the request with a “File not found” message.

Note For a procedure to enable TFTP server or gateway, see Chapter 7 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

Note The **trusted-host** command has been deprecated in the ACNS software, Release 5.2 and later releases and cannot be used to permit or deny access to TFTP services.

In addition to enabling the TFTP server, you must perform some additional steps depending on the functionality that you want to use. These additional steps are described in the following sections.

Serving Local Content

To serve requests for local content, you must enable the TFTP server and also configure the local directories to use. This functionality is available on any ACNS device, including a Content Engine, Content Router, Content Distribution Manager, or IP/TV Program Manager. To configure the local TFTP directories, follow these steps:

1. Create the local directories using the **mkdir** command.
2. 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.

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.

Serving the Pre-Positioned Content

To serve the pre-positioned content, you must enable the TFTP server and also identify the remote server from which the files have been pre-positioned. This functionality is available only on a Content Engine.

To identify where the files have been pre-positioned, use the **tftp-server gw proto server** *hostname* **path** *directory* **command**. Replace *hostname* with the name of the origin server from which the pre-positioned files are acquired. Replace *directory* with the location where the files were pre-positioned through the acquisition process.

Using the TFTP Gateway to Serve Content from Remote Servers

To serve the content from remote servers, you must enable the TFTP server and also identify the remote servers to which the TFTP gateway will direct requests. This functionality is available only on a Content Engine.

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**. To complete this command, identify the protocol (HTTP or FTP), the hostname or IP address of the server, and any authentication information required to access the remote server**.** You can enter the **tftp-server gw proto** command twice to configure a primary and a backup HTTP or FTP server. Use the **priority** option to specify whether the server is primary (priority = 1) or backup (priority = 2).

Note For information on how the TFTP server and gateway work, see Chapter 7 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

Using IP ACLs to Control TFTP Access

You can enable TFTP applications to be attached to a particular interface (such as management services to the private IP address space) so that the Content Engine can have one interface in the customer's IP address space that serves the content, and another interface in a private IP address space that the administrator uses for management purposes. This setting ensures that clients can access the Content

Engine only in the public IP address space for serving content and not access it for management purposes. A device attempting to access one of these applications associated with an access control list (ACL) must be on the list of trusted devices to be allowed access.

Examples

The following example enables the TFTP server:

```
ContentEngine(config) inetd enable tftp
```

The following example defines an access list that permits access to the TFTP service for TFTP 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
```

The following example configures two local directories from which the Content Engine will try to fulfill TFTP requests:

```
ContentEngine(config)# mkdir /local1/mydir
ContentEngine(config)# mkdir /local1/clients
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.

The following example identifies the IP access list that permits access to the TFTP service:

```
ContentEngine(config)# tftp-server access-list 1
```

The following example identifies an FTP server to which the Content Engine will forward requests when it cannot find the file in its local directories:

```
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 caching application to obtain the file from the remote server. The URL created by using this example configuration would be as follows:

ftp://myuser:mypasswd@192.168.100.1/myremotedir/*requested-file-name*

The following example identifies the remote server from which files have been pre-positioned and allows the TFTP gateway to serve TFTP requests from the pre-positioned content:

```
ContentEngine(config)# tftp-server gw proto http 192.168.100.2 pri 1 path /myremotedir name
myuser passwd mypassword
```

The URL formed by using this example configuration would be as follows:

http://myuser:mypasswd@192.168.100.2/myremotedir/*requested-file-name*

In this case, the caching application searches for the requested file in the pre-positioned directory. If the requested file is not found, the caching application makes a request to the remote server that is specified by the URL.

Related Commands

debug http header
debug http all
debug tftp-server
inetd enable tftp
ip access-list
mkdir

show inetd
show statistics tftp
show tftp-server

traceroute

To trace the route to a remote host, use the **traceroute** EXEC command.

traceroute {*hostname* | *ip-address*}

Syntax Description

hostname	Name of the remote host.
ip-address	IP address of the remote host.

Defaults

No default behavior values

Command Modes

EXEC

Usage Guidelines

Traceroute is a widely available utility on most operating systems. Similar to ping, traceroute is a valuable tool for determining connectivity in a network. Ping allows the user to find out if there is a connection between the two end systems. Traceroute does this as well, but additionally lists the intermediate routers between the two systems. Users can see the routes that packets can take from one system to another. Use the **traceroute** command to find the route to a remote host when either the hostname or the IP address is known.

The **traceroute** command uses the TTL field in the IP header to cause routers and servers to generate specific return messages. Traceroute starts by sending a UDP datagram to the destination host with the TTL field set to 1. If a router finds a TTL value of 1 or 0, it drops the datagram and sends back an ICMP time-exceeded message to the sender. The traceroute facility determines the address of the first hop by examining the source address field of the ICMP time-exceeded message.

To identify the next hop, traceroute sends a UDP packet with a TTL value of 2. The first router decrements the TTL field by 1 and sends the datagram to the next router. The second router sees a TTL value of 1, discards the datagram, and returns the time-exceeded message to the source. This process continues until the TTL is incremented to a value large enough for the datagram to reach the destination host (or until the maximum TTL is reached).

To determine when a datagram has reached its destination, traceroute sets the UDP destination port in the datagram to a very large value that the destination host is unlikely to be using. When a host receives a datagram with an unrecognized port number, it sends an ICMP “port unreachable” error to the source. This message indicates to the traceroute facility that it has reached the destination.

Examples

The following example shows how to trace the route to a remote host from the Content Engine:

```
CONTENTENGINE# traceroute 10.77.157.43
traceroute to 10.77.157.43 (10.77.157.43), 30 hops max, 38 byte packets
 1  10.1.1.50 (10.1.1.50)  2.024 ms  2.086 ms  2.219 ms
 2  sblab2-rtr.cisco.com (192.168.10.1)  3.718 ms 172.19.231.249 (172.19.231.249)  0.653
ms  0.606 ms
 3  sjc22-00lab-gw1.cisco.com (172.24.115.65)  0.666 ms  0.624 ms  0.597 ms
 4  sjc20-lab-gw2.cisco.com (172.24.115.109)  0.709 ms  0.695 ms  0.616 ms
 5  sjc20-sbb5-gw2.cisco.com (128.107.180.97)  0.910 ms  0.702 ms  0.674 ms
 6  sjc20-rbb-gw5.cisco.com (128.107.180.9)  0.762 ms  0.702 ms  0.664 ms
 7  sjc12-rbb-gw4.cisco.com (128.107.180.2)  0.731 ms  0.731 ms  0.686 ms
```

```
 8  sjc5-gb3-f1-0.cisco.com (10.112.2.158)  1.229 ms  1.186 ms  0.753 ms
 9  capnet-hkidc-sjc5-oc3.cisco.com (10.112.2.238)  146.784 ms  147.016 ms  147.051 ms
10  hkidc-capnet-gw1-g3-1.cisco.com (10.112.1.250)  147.163 ms  147.319 ms  148.050 ms
11  hkidc-gb3-g0-1.cisco.com (10.112.1.233)  148.137 ms  148.332 ms  148.361 ms
12  capnet-singapore-hkidc-oc3.cisco.com (10.112.2.233)  178.137 ms  178.273 ms  178.005
ms
13  singapore-capnet2-fa4-0.cisco.com (10.112.2.217)  179.236 ms  179.606 ms  178.714 ms
14  singapore-gb1-fa2-0.cisco.com (10.112.2.226)  179.499 ms  179.914 ms  179.873 ms
15  capnet-chennai-singapore-ds3.cisco.com (10.112.2.246)  211.858 ms  212.167 ms  212.854
ms
16  hclodc1-rbb-gw2-g3-8.cisco.com (10.112.1.213)  213.639 ms  212.580 ms  211.211 ms
17  10.77.130.18 (10.77.130.18)  212.248 ms  212.478 ms  212.545 ms
18  codc-tbd.cisco.com (10.77.130.34)  212.315 ms  212.688 ms  213.063 ms
19  10.77.130.38 (10.77.130.38)  212.955 ms  214.353 ms  218.169 ms
20  10.77.157.9 (10.77.157.9)  217.217 ms  213.424 ms  222.023 ms
21  10.77.157.43 (10.77.157.43)  212.750 ms  217.260 ms  214.610 ms
```

The following example shows how the **traceroute** command fails to trace the route to a remote host from the Content Engine:

```
CONTENTENGINE# traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 38 byte packets
 1  10.1.1.50 (10.1.1.50)  2.022 ms  1.970 ms  2.156 ms
 2  sblab2-rtr.cisco.com (192.168.10.1)  3.955 ms 172.19.231.249 (172.19.231.249)  0.654
ms  0.607 ms
 3  sjc22-00lab-gw1.cisco.com (172.24.115.65)  0.704 ms  0.625 ms  0.596 ms
 4  sjc20-lab-gw1.cisco.com (172.24.115.105)  0.736 ms  0.686 ms  0.615 ms
 5  sjc20-sbb5-gw1.cisco.com (128.107.180.85)  0.703 ms  0.696 ms  0.646 ms
 6  sjc20-rbb-gw5.cisco.com (128.107.180.22)  0.736 ms  0.782 ms  0.750 ms
 7  sjce-rbb-gw1.cisco.com (171.69.7.249)  1.291 ms  1.314 ms  1.218 ms
 8  sjce-corp-gw1.cisco.com (171.69.7.170)  1.477 ms  1.257 ms  1.221 ms
 9  * * *
10  * * *
.
.
.
29  * * *
30  * * *
```

Table 2-173 describes the fields in the **traceroute** command output.

Table 2-173 traceroute Command Output Fields

Field	Description
30 hops max, 38 byte packets	Maximum TTL value and the size of the ICMP datagrams being sent.
2.022 ms 1.970 ms 2.156 ms	Total time (in milliseconds) for each ICMP datagram to reach the router or host plus the time it took for the ICMP time-exceeded message to return to the host. An exclamation point following any of these values (for example, 20 ms !) indicates that the port-unreachable message returned by the destination had a TTL of 0 or 1. Typically, this situation occurs when the destination uses the TTL value from the arriving datagram as the TTL in its ICMP reply. The reply does not arrive at the source until the destination receives a traceroute datagram with a TTL equal to the number of hops between the source and destination.
*	An asterisk (*) indicates that the timeout period (default of 5 seconds) expired before an ICMP time-exceeded message was received for the datagram.

Related Commands

ping

transaction-log force

To force the archive or export of the transaction log, use the **transaction-log force** EXEC command.

transaction-log force {**archive** | **export**}

Syntax Description

archive	Forces the archive of the *working.log* file.
export	Forces the archived files to be exported to an FTP server.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

The **transaction-log force archive** command causes the transaction log *working.log* file to be archived to the Content Engine hard disk following the next transaction. This command has the same effect as the **clear transaction-log** command.

The **transaction-log force export** command causes the transaction log to be exported to an FTP server designated by the **transaction-logs export ftp-server** command.

The **transaction-log force** commands do not change the configured or default schedule for archive or export of transaction log files. If the archive interval is configured in seconds or the export interval is configured in minutes, the forced archive or export interval period is restarted after the forced operation.

If a scheduled archive or export job is in progress when a corresponding **transaction-log force** command is entered, the command has no effect. If a **transaction-log force** command is in progress when an archive or export job is scheduled to run, the forced operation is completed and the archive or export is rescheduled for the next configured interval.

Examples

The following example shows how to archive the transaction log file to the Content Engine hard disk:

```
ContentEngine# transaction-log force archive
```

The following example shows that 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
```

The following example shows how to export the transaction log file from the Content Engine hard disk to an FTP server designated by the **transaction-logs export ftp-server** command:

```
ContentEngine# transaction-log force export
```

Related Commands

clear statistics transaction-logs
clear transaction-log
show statistics transaction-logs
show transaction-logging
transaction-logs

transaction-logs

To configure and enable transaction logs, use the **transaction-logs** command in global configuration mode. To disable transaction logs, use the **no** form of this command.

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*]

transaction-logs archive max-file-size *filesize*

transaction-logs enable

transaction-logs export compress

transaction-logs export enable

transaction-logs export ftp-server {*hostname* | *servipaddrs*} *login passw directory*

transaction-logs export interval *minutes*

transaction-logs export interval every-day {**at** *hour:minute* | **every** *hours*}

transaction-logs export interval every-hour {**at** *minute* | **every** *minutes*}

transaction-logs export interval every-week [**on** *weekdays* **at** *hour:minute*]

transaction-logs export sftp-server {*hostname* | *servipaddrs*} *login passw directory*

transaction-logs file-marker

transaction-logs format {**apache** | **custom** *string* | **extended-squid** | **squid**}

transaction-logs log-windows-domain

transaction-logs logging {**enable** | **entry-type** {**all** | **request-auth-failures**} | **facility** {*parameter*} | **host** {*hostname* | *ip-address* [**port** *port-num* | **rate-limit** *rate*]}}

transaction-logs sanitize

no transaction-logs {**archive** {**interval** | **max-file-size**} | **enable** | **export** {**compress** | **enable** | **ftp-server** {*hostname* | *servipaddrs*} | **interval** | **sftp-server** {*hostname* | *servipaddrs*}} | **file-marker** | **format** | **log-windows-domain** | **logging** {**enable** | **entry-type** | **facility** | **host** {*hostname* | *ip-address*} [**port** *port-number*] [**rate-limit** *rate*]} | **sanitize**}

Syntax Description

archive	Configures archive parameters.
interval	Determines how frequently the archive file is to be saved.
seconds	Frequency of archiving in seconds (120–604800).
every-day	Archives using intervals of 1 day or less.

at	Specifies the local time at which to archive each day.
hour:minute	Time of day at which to archive in local time (hh:mm).
every	Specifies the interval in hours. Interval aligns with midnight.
hours	Number of hours for daily file archive. 1 Hourly 12 Every 12 hours 2 Every 2 hours 24 Every 24 hours 3 Every 3 hours 4 Every 4 hours 6 Every 6 hours 8 Every 8 hours
every-hour	Specifies the archives using intervals of 1 hour or less.
at	Sets the time to archive at each hour.
minute	Minute alignment for the hourly archive (0–59).
every	Specifies the interval in minutes for hourly archive that aligns with the top of the hour.
minutes	Number of minutes for hourly archive. 10 Every 10 minutes 15 Every 15 minutes 2 Every 2 minutes 20 Every 20 minutes 30 Every 30 minutes 5 Every 5 minutes
every-week	Archives using intervals of 1 or more times a week.
on	(Optional) Sets the day of the week on which to archive.
weekdays	Weekdays on which to archive. One or more weekdays can be specified. Fri Every Friday Mon Every Monday Sat Every Saturday Sun Every Sunday Thu Every Thursday Tue Every Tuesday Wed Every Wednesday
at	(Optional) Sets the local time at which to archive each day.
hour:minute	Time of day at which to archive in local time (hh:mm).
max-file-size	Sets the maximum archive file size.
filesize	Maximum archive file size in kilobytes (1000–2000000).
enable	Enables the transaction log.
export	Configures file export parameters.
compress	Compresses the archived files in the gzip format before exporting.
enable	Enables the exporting of log files at the specified interval.
ftp-server	Sets the FTP server to receive exported archived files.
hostname	Hostname of the target FTP server.
servipaddrs	IP address of the target FTP server.

login	User login to target FTP server.
passw	User password to target FTP server.
directory	Target directory path for exported files on FTP server.
interval	Determines how frequently the file is to be exported.
minutes	Number of minutes in the interval at which to export a file (1–10080).
every-day	Specifies the exports using intervals of 1 day or less.
at	Specifies the local time at which to export each day.
hour:minute	Time of day at which to export in local time (hh:mm).
every	Specifies the interval in hours for the daily export.
hours	Number of hours for the daily export. 1 Hourly 12 Every 12 hours 2 Every 2 hours 24 Every 24 hours 3 Every 3 hours 4 Every 4 hours 6 Every 6 hours 8 Every 8 hours
every-hour	Specifies the exports using intervals of 1 hour or less.
at	Specifies the time at which to export each hour.
minute	Minute (0–59) alignment for the hourly export.
every	Specifies the interval in minutes that align with the top of the hour.
minutes	Number of minutes for the hourly export. 10 Every 10 minutes 15 Every 15 minutes 2 Every 2 minutes 20 Every 20 minutes 30 Every 30 minutes 5 Every 5 minutes
every-week	Specifies the exports using intervals of 1 of more times a week.
on	(Optional) Specifies the days of the week for the export.
weekdays	Weekdays on which to export. One or more weekdays can be specified. Fri Every Friday Mon Every Monday Sat Every Saturday Sun Every Sunday Thu Every Thursday Tue Every Tuesday Wed Every Wednesday
at	(Optional) Specifies the time of day at which to perform the weekly export.
hour:minute	Time of day at which to export in the local time (hh:mm).
sftp-server	Sets the Secure File Transfer Protocol (SFTP) server to receive exported archived files.
hostname	Hostname of the target SFTP server.
servipaddrs	IP address of the target SFTP server.

login	User login to the target SFTP server (less than 40 characters).
passw	User password to the target SFTP server (less than 40 characters).
directory	Target directory path for exported files on the SFTP server.
file-marker	Adds statements to the transaction log indicating the file beginning and end.
format	Sets the format to use for the HTTP transaction log entries in the working.log file.
apache	Configures the HTTP transaction logs output to the Apache Common Log format (CLF).
custom	Configures the HTTP transaction logs output to the custom log format.
string	Quoted log format string containing the custom log format.
extended-squid	Configures the HTTP transaction logs output to the Extended Squid log format.
squid	Configures the HTTP transaction logs output to the Squid native log format (Squid–1.1 *access.log* format).
log-windows-domain	Logs the Windows domain with an authenticated username if available in HTTP transaction log entries.
logging	Configures the Content Engine to send HTTP remote transaction log messages to a remote syslog server.
enable	Enables the remote transaction logging.
entry-type	Specifies the type of transaction log entry.
all	Sets the Content Engine to send all transaction log messages to the remote syslog server.
request-auth-failures	Sets the Content Engine to log to the remote syslog server only those transactions that the Content Engine failed to authenticate with the authentication server. **Note** Only those authentication failures that are associated with an end user who is attempting to contact the authentication server are logged. The transactions in pending state (that have contacted the authentication server, but waiting for a response from the authentication server) are not logged.
facility	Configures a unique facility to create a separate log on the remote syslog host for real-time transaction log entries.

parameter	Specifies one of the following facilities: **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
host	Configures the remote syslog server.
hostname	Hostname of the remote syslog server.
ip-address	IP address of the remote syslog server.
port	Configures the port to use when sending transaction log messages to the syslog server.
port-num	Port number to use when sending transaction log messages to the syslog server (the default is 514).
rate-limit	Configures the rate at which the transaction logger is allowed to send messages to the remote syslog server.
rate	Rate (number of messages per second) at which the transaction logger is allowed to send messages to the remote syslog server.
sanitize	Writes user IP addresses in the log file as 0.0.0.0 in HTTP transaction log entries.

Defaults

archive: disabled

enable: disabled

export compress: disabled

export: disabled

file-marker: disabled

sanitize: disabled

archive interval: every day, every one hour

archive max-file-size: 2,000,000 KB

export interval: every day, every one hour

format: Squid native log format

logging port *port-num*: 514

Command Modes global configuration

Usage Guidelines Content Engines that are running the ACNS 5.x software can record all errors and access activities. In the ACNS 5.x software, each content service module 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 used for problem identification and solving, load monitoring, billing, statistical analysis, security problems, and cost analysis and provisioning.

The translog module on the Content Engine handles transaction logging and supports the Apache Common Log Format (CLF), Squid format, Extended Squid format, and the World Wide Web Consortium (W3C) customizable logging format.

Note In the ACNS 5.3 software release and later releases, you cannot configure invalid custom log format strings. However, releases of ACNS software prior to Release 5.3 allow you to configure invalid custom log format strings. When you upgrade the ACNS devices from the ACNS 5.2 software to the ACNS 5.3 software, any invalid custom log formats that had been configured are deleted.

Note 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 situation occurs mostly with the WMT RTSPU protocol due to the behavior of the player.

Enable transaction log recording with the **transaction-logs enable** command. The transactions that are logged include HTTP and FTP. In addition, ACNS 5.5 software supports Extensible Markup Language (XML) logging for MMS-over-HTTP and MMS-over-RTSP (RTSP over Windows Media Services 9).

When enabled, daemons create a *working.log* file in /local1/logs/ on the sysfs volume for HTTP and FTP transactions and a separate *working.log* file in /local1/logs/export for Windows Media transactions. 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.

The *working.log* file is a link to the actual log file with the timestamp embedded in its filename. When you configure the **transaction-logs archive interval** command, the first transaction that arrives after the interval elapses is logged to the *working.log* file as usual, and then actual log file is archived and a new log file is created. Only transactions subsequent to the archiving event are recorded in the new log file. The *working.log* file is then updated to point to the newly created log file. The transaction log archive file naming conventions are shown in Table 2-177. The Content Engine default archive interval is once an hour every day.

Use the **transaction-logs archive max-file-size** command to specify the maximum size of an archive file. The *working.log* file is archived when it attains the maximum file size if this size is reached before the configured archive interval time.

Use the **transaction-logs file-marker** option to mark the beginning and end of the HTTP, HTTPS, and FTP proxy logs. By examining the file markers of an exported archive file, you can determine whether the FTP process transferred the entire file. The file markers are in the form of dummy transaction entries that are written in the configured log format.

The following example shows the start and end dummy transactions in the default native Squid log format.

- 970599034.130 0 0.0.0.0 TCP_MISS/000 0 NONE TRANSLOG_FILE_START - NONE/- -
- 970599440.130 0 0.0.0.0 TCP_MISS/000 0 NONE TRANSLOG_FILE_END - NONE/- -

Use the **format** option to format the HTTP, HTTPS, and FTP proxy log files for custom format, native Squid or Extended Squid formats, or Apache Common Log Format (CLF).

The **transaction-logs format custom** command allows you to use a *log format string* to log additional fields that are not included in the predefined native Squid or Extended Squid formats or the Apache CLF format. The log format string is a string that contains the tokens listed in Table 2-174 and mimics the Apache log format string. The log format string can contain literal characters that are copied into the log file. Two backslashes (\\) can be used to represent a literal backslash, and a backslash followed by a single quotation mark (\') can be used to represent a literal single quotation mark. A literal double quotation mark 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.

Table 2-174 lists the acceptable format tokens for the log format string. The ellipsis (...) portion of the format tokens shown in this table represent 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 2-174 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 hyphen (-). 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, which means that the value associated with the format token is logged if none of the status codes listed after the exclamation point (!) match the HTTP status code of the request. If any of the status codes listed after the exclamation point (!) match the HTTP status code of the request, then a hyphen (-) 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, and %!200,304,302{Referer}i logs the Referer header value on all requests that did not return a normal status.

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:

%r

%{Referer}i

%{User-Agent}i

The %{Cookie}i format token is generated without the surrounding double quotation marks, because the Cookie value can contain double quotes. The Cookie value can contain multiple attribute-value pairs that are separated by spaces. We recommend that when you use the Cookie format token in a custom format string, you should position it as the last field in the format string so that it can be easily parsed by the transaction log reporting tools. By using the format token string \'%{Cookie}i\' the Cookie header can be surrounded by single quotes (').

The following command can generate the well-known Apache Combined Log Format:

transaction-log 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 transaction log entry example in the Apache Combined Format is configured using the preceding custom format string:

```
[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)"
```

Table 2-174 Custom Format "Log Format String" Values

Format Token	Value
%...a	IP address of the requesting client.
%...A	IP address of the Content Engine.
%...B %...b	Bytes sent excluding HTTP headers.
%...c	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.
%...f	Filename.
%...h	Remote host (IP address of the requesting client is logged).
%...H	Request protocol.
%...{Foobar}i	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.
%...l	Remote log name. Not implemented on the Content Engine, so a hyphen (-) is logged.
%...m	Request method.
%...p	Canonical port of the server servicing the request. Not applicable on the Content Engine, so a hyphen (-) is logged.
%...P	Process ID of the child that serviced the request.
%...q	Query string (that is preceded by a question mark (?) if a query string exists; otherwise, it is an empty string).
%...r	First line of the request.
%...s	Status. The translog code always returns the HTTP response code for the request.
%...t	Time in common log time format (or standard English format).
%...{format}t	Time in the form given by the format token specified in Table 2-175.
%...T	Time consumed to serve the request in seconds (a floating point number with three decimal places).
%...u	Remote user.
%...U	URL path requested not including query strings.
%...v %...V	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.

Table 2-175 specifies the format token for the date and time of the format token %...{format}t that is listed in Table 2-176.

Table 2-175 Format Token for Date and Time

Format Token	Value
%a	Abbreviated weekday name.
%A	Full weekday name.
%b	Abbreviated month name.
%B	Full month name.
%c	Date and time representation.
%C	Century number (year/100) as a 2-digit integer.
%d	Day of the month as a decimal number (01–31).
%D	Equivalent to %m/%d/%y. (In countries other than the USA, %d/%m/%y is common. In an international context, this format is ambiguous and should not be used.)
%e	Similar to %d, the day of the month as a decimal number, but a leading zero is replaced by a space.
%G	ISO 8601 year with the century as a decimal number. The 4-digit year corresponding to the ISO week number (see %V). This format token 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.
%g	Similar to %G, but without a century; that is, with a 2-digit year (00–99).
%h	Equivalent to %b.
%H	Hour as a decimal number using a 24-hour clock (00–23).
%I	Hour as a decimal number using a 12-hour clock (01–12).
%j	Day of the year as a decimal number (001–366).
%k	Hour (24-hour clock) as a decimal number (0–23); single digits are preceded by a blank. (See also %H.)
%l	Hour (12-hour clock) as a decimal number (1–12); single digits are preceded by a blank. (See also %I.)
%m	Month as a decimal number (01–12).
%M	Minute as a decimal number (00–59).
%n	New line character.
%p	Either AM or PM according to the given time value, or the corresponding strings for the current locale. Noon is treated as PM and midnight as AM.
%P	Similar to %p but in lowercase: am or pm or a corresponding string for the current locale.
%r	Time in a.m. or p.m. notation. This format token is equivalent to "%I:%M:%S %p."
%R	Time in 24-hour notation (%H:%M). For a version including the seconds, see %T below.
%s	Number of seconds since the epoch; that is, since 1970-01-01 00:00:00 UTC.
%S	Second as a decimal number (00–61).
%t	Tab character.
%T	Time in 24-hour notation (%H:%M:%S).

Table 2-175 Format Token for Date and Time (continued)

Format Token	Value
%u	Day of the week as a decimal, 1–7, Monday being 1. See also %w.
%U	Week number of the current year as a decimal number (00–53), starting with the first Sunday as the first day of week 01. See also %V and %W.
%V	ISO 8601:1988 week number of the current year as a decimal number (01–53), where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. See also %U and %W.
%w	Day of the week as a decimal (0–6) with Sunday as 0. See also %u.
%W	Week number of the current year as a decimal number (00–53), starting with the first Monday as the first day of week 01.
%x	Date representation without the time.
%X	Time representation without the date.
%y	Year as a decimal number without a century (00–99).
%Y	Year as a decimal number, including the century.
%z	Time zone as an hour offset from GMT. Required to emit RFC822-conformant dates (using %a, %d %b %Y %H:%M:%S %z).
%Z	Time zone or name or abbreviation.
%%	Literal % character.

The Extended Squid log format uses the RFC 981 field of the Squid log format for the username. The Extended Squid format logs the associated username for authentication for each record in the log file, if available. The username is also used for billing purposes.

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. Additional format tokens that are extensions to the W3C Customized Logging Format were added in the ACNS 5.3 software release to support additional Cisco and Squid customized logging fields. These new format tokens provide support for a Squid-like logging format from within the W3C customizable token set.

In the ACNS 5.3 software release, the following transaction logging support was added:

- Support for the Extended Squid-equivalent internal 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 software release, the W3C Customizable Logging Format was extended to include support for the following special token sequence:

%...{<translog-token>}C

The ellipsis (...) is optional. If specified, it can be a sequence of conditional HTTP response codes separated by commas. The uppercase C 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 2-176 lists the existing and new <translog-token> directives from the Extended Squid format, which are not immediately supported by the W3C definitions but are supported in the ACNS 5.3 software and later releases.

Table 2-176 Translog Token Directives

Format Token	Value
%...{es}C	Current time presented as the number of seconds that have elapsed since the Epoch (Jan. 1st. 1970).
%...{em}C	Current number of milliseconds that have elapsed since the Epoch (Jan. 1st. 1970).
%...{te}C	Number of milliseconds that have elapsed until the request was completed.
%...{rd}C	Squid-like cache-status code string (for example, TCP_HIT and TCP_CLIENT_REFRESH_MISS).
%..{cs}C	Number of bytes sent to the client (including the protocol headers).
%...{rh}C	Strict Squid-style hierarchy as it applies to the Content Engine.
%...{rh}CE	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_addess* is logged instead of the DIRECT/*origin_server_ip_address*.
%...{rt}C	MIME type of the object in the response, as specified by any protocol headers that define such MIME types. Currently, the ACNS 5.3 software does not support logging the MIME type of the object that is being requested, and a hyphen (-) is logged instead. **Note** This restriction also applies to the Squid and Extended Squid logging formats. **Tip** A MIME-type association 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. You can edit, add, or delete these MIME-type association rules in the browsers. For example, through a 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.
%...{ru}C	URL being requested including any additional query strings.
%...{as}C	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 logs information where this token sequence is used.

In addition to the tokens listed in Table 2-176, you can condense multiple %...{xx}C style tokens into a single embedded token sequence within the %...{xx}C style. A limited customized logging string validation mechanism has been implemented for all the %...{xy}C style format tokens. This mechanism ensures that the tokens are valid and rejects invalid tokens. 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 percent (%) symbol as follows:

%{rh}C %{rt}C %{as}C

can be reexpressed in a condensed embedded token format as the following:

%{%rh %rt %as}C

The command line syntax accepts single tokens represented as the following:

%{%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 above set of tokens allow you to configure an extended Squid-like format line within the W3C Customizable Logging format specification as follows:

%{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 a 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 logged within the log file as the characters that made up the token. For example, %{xy}C is logged into the log file as xy. All characters outside of a token specification sequence are repeated verbatim within the log file.

Sanitizing Transaction Logs

Use the **sanitized** option to disguise the IP address 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.

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 (configured with the **transaction-logs format custom** *string* global configuration command in which the 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 could export transaction logs to an external host.

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

The log files automatically have the following 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 WMT logs have no .txt extension in the filename.

Exporting and Archiving Intervals

The transaction log archive and export functions are configured with the following commands:

- The **transaction-logs archive interval** global configuration command allows the administrator to specify when the *working.log* file is archived.
- The **transaction-logs export interval** global configuration command allows the administrator to specify when the archived transaction logs are exported.

The following limitations apply:

- When the interval is scheduled in units of hours, the value must divide evenly into 24. For example, the interval can be every 4 hours, but not every 5 hours.
- When the interval is scheduled in units of minutes, the value must divide evenly into 60.
- Only the more common choices of minutes are supported. For example, the interval can be 5 minutes or 10 minutes, but not 6 minutes.
- The selection of interval alignment is limited. If an interval is configured for every 4 hours, it will align with midnight. It cannot align with 12:30 or with 7 a.m.
- The feature does not support different intervals within a 24-hour period. For example, it does not support an interval that is hourly during regular business hours and then every 4 hours during the night.

Transaction Log Archive Filenaming Convention

The archive transaction log file is named as follows for HTTP and WMT caching:

```
celog_10.1.118.5_20001228_235959.txt

mms_export_10.1.118.5_20001228_235959
```

If the **export compress** feature is enabled when the file is exported, then the file extension will be *.gz* after the file is compressed for the export operation, as shown in the following example:

```
celog_10.1.118.5_20001228_235959.txt.gz

mms_export_10.1.118.5_20001228_235959.gz
```

Table 2-177 describes the name elements.

Table 2-177 Description of Archive Log Name Elements

Sample of Element	Description
acqdist_	Acquisition and distribution archive log file.
celog_	HTTP caching proxy server archive file.
cifs_server_	Windows file sharing server archive file.
cseaccess	Cisco Streaming Engine archive file.
icap_	ICAP server archive file.
mms_export_	Standard Windows Media Services 4.1 caching proxy server archive file.
mms_export_e_wms_41_	Extended Windows Media Services 4.1 caching proxy server archive file.
mms_export_wms_90_	Standard Windows Media Services 9.0 caching proxy server archive file.
mms_export_e_wms_90_	Extended Windows Media Services 9.0 caching proxy server archive file.
rproxyaccess.log.	RealProxy archive file.

Table 2-177 Description of Archive Log Name Elements (continued)

Sample of Element	Description
rmsvraccess.log.	RealSubscriber archive file.
tftp_server_	TFTP server archive file.
tvout_	TV-out program archive file.
10.1.118.5_	IP address of the Content Engine creating the archive file.
20001228_	Date on which the archive file was created (yyyy/mm/dd).
235959	Time when the archive file was created (hh/mm/ss).

Table 2-178 lists the directory names and the corresponding examples of the archive filenames.

Table 2-178 Archive Filename Examples and Directories

Directory	Archive Filename
logs	celog_10.1.94.4_20050310_231500.txt
logs/export	mms_export_10.1.94.4_20050315_001545
logs/export/extended-wms-41	mms_export_e_wms_41_10.1.94.4_20050315_001545
logs/wms-90	mms_export_wms_90_10.1.94.4_20050315_001545
logs/export/extended-wms-90	mms_export_e_wms_90_10.1.94.4_20050315_001545
logs/acqdist	acqdist_10.1.94.4_20050315_001545
logs/cifs_server	cifs_server_10.1.94.4_20050315_001545
logs/cisco-streaming-engine	cseaccess10.1.94.4__050315000.log
logs/icap	icap_10.1.94.4_20050315_001545
logs/real-proxy	rproxyaccess.log.10.1.94.4_.20050315_001545
logs/real-subscriber	rmsvraccess.log.10.1.94.4_.20050315_001545
logs/tftp_server	tftp_server_10.1.94.4_20050315_001545
logs/tvout	tvout_10.1.94.4_20050315_001545

Compressing Archive Files

The **transaction-logs export compress** option compresses an archive into a gzip file format before exporting it. Compressing the archive file uses less disk space on both the Content Engine and the FTP export server. The compressed file uses less bandwidth when transferred. The archive filename of the compressed file has the extension *.gz*.

Exporting Transaction Logs to External FTP Servers

The **transaction-logs export ftp-server** option can support up to four FTP servers. To export transaction logs, you must first enable the feature and configure the FTP server parameters. The following information is required for each target FTP server:

- FTP server IP address or the hostname

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

- FTP user login and user password
- Path of the directory where transferred files are written

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

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

Exporting Transaction Logs to External SFTP Servers

Use the **transaction-logs export sftp-server** option to export transaction logs. You must first enable the feature and configure the Secure File Transfer Protocol (SFTP) server parameters. The following information is required for each target SFTP server:

- SFTP server IP address or the hostname

  The Content Engine translates the hostname with a DNS lookup and then stores the IP address in the configuration.
- SFTP user login and user password
- Path of the directory where transferred files are written

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

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

Receiving a Permanent Error from the External FTP Server

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.

When an FTP server returns a permanent error to the Content Engine, the export is retried at 10-minute intervals or sooner if the configured export interval is sooner. If the error is a result of a misconfiguration of the **transaction-logs export ftp server** command, then you must reenter the Content Engine parameters to clear the error condition. The **show statistics transaction-logs** command displays the status of logging attempts to export servers.

The **show statistics transaction-logs** command shows that the Content Engine failed to export archive files.

The **transaction-logs format** command has four options: **squid**, **extended-squid**, **apache**, and **custom**. The default format is **squid**.

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

Configuring Intervals Between 1 Hour and 1 Day

The archive or export interval can be set for once a day with a specific time stamp. It can also be set for hour frequencies that align with midnight. For example, every 4 hours means archiving occurs at 0000, 0400, 0800, 1200, and 1600. It is not possible to archive at half-hour intervals such as 0030, 0430, or 0830. The following intervals are acceptable: 1, 2, 3, 4, 6, 8, 12, and 24.

Configuring Intervals of 1 Hour or Less

The interval can be set for once an hour with a minute alignment. It can also be set for frequencies of less than an hour; these frequencies will align with the top of the hour. Every 5 minutes means that archiving will occur at 1700, 1705, and 1710.

Configuring Export Interval on Specific Days

The export interval can be set for specific days of the week at a specific time. One or more days can be specified. The default time is midnight.

You must be aware that archived logs are automatically deleted when free disk space is low. It is important to select an export interval that exports files frequently enough so that files are not automatically removed prior to export.

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 hyphen (-) 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 domain\username format 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 as 1.)

Use the **no transaction-logs log-windows-domain** global configuration command to negate logging NTLM parameters to the transaction log in Extended Squid-style or Custom formats.

Monitoring HTTP Request Authentication Failures in Real Time

The ACNS 5.2 software and later releases support sending HTTP transaction log messages to a remote syslog server so that you can monitor the remote syslog server for HTTP request authentication failures in real time. This real-time transaction log 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 Because system logging (syslog) occurs through UDP, the message transport to the remote syslog host is not reliable.

To support the real-time transaction logging feature, the following commands are supported in the ACNS 5.2 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 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.

Configuring the Remote syslog Host for Real-Time Transaction Logging

To configure a Content Engine to send transaction log messages in real time to a remote syslog host, use the **transaction-logs logging host** global configuration command.

The ACNS 5.2 software and later releases support Content Engine syslog messages 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

When logging transactions to a remote syslog host, use the **transaction-logs logging facility** global configuration command to configure a transaction log facility.

Specifying the Transaction Log Entry Type when Logging to a Remote syslog Host

With the ACNS 5.2 software and later releases, use the **transaction-logs logging entry-type** global configuration command to configure the Content Engine to send only the transactions associated with HTTP request authentication failures or to send all of the transactions.

When you specify the **transaction-logs logging enable** global configuration command, 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.

Examples

The following example shows how to configure an FTP server:

```
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
```

The following example shows how to delete an FTP server:

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

Use the **no** form of the command to disable the entire transaction log export feature while retaining the rest of the configuration:

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

The following example shows how to change a username, password, or directory:

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

Note For security reasons, passwords are never displayed.

The following example shows how to restart the export of archive transaction logs:

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

The following example shows how to delete an SFTP server from the current configuration:

```
ContentEngine(config)# no transaction-logs export sftp-server sftphostname
```

The following example shows how to display information on exported log files:

```
ContentEngine# show transaction-logging
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
Exporting files to ftp servers is enabled.
File compression is disabled.
Export interval: every-day every 1 hour.
ftp-server           username        directory
10.1.1.1              mylogin        /ftpdirectory
10.2.2.2              mylogin        /ftpdirectory
 Working Log file - size: 103
                     age: 0
```

The following examples show how to configure the archiving intervals:

```
ContentEngine(config)# transaction-logs archive interval every-day
  at        Specify the time at which to archive each day
  every     Specify the interval in hours. It will align with midnight

ContentEngine(config)# transaction-logs archive interval every-day at
<0-23>:  Time of day at which to archive (hh:mm)

ContentEngine(config)# transaction-logs archive interval every-day every
<1-24>  Interval in hours: {1, 2, 3, 4, 6, 8, 12 or 24}
```

The following example shows that the Content Engine has failed to export archive files:

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

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

The following example shows how to correct a misconfiguration:

```
ContentEngine(config)# transaction-logs export ftp-server 10.1.1.1 goodlogin pass
/ftpdirectory
```

The working.log file and archived log files are listed for HTTP and WMT.

The following example shows how to export transaction logs to an SFTP server:

```
ContentEngine(config)# transaction-logs export sftp-server 10.1.1.100 mylogin mypasswd
/mydir
```

The following example shows how to archive every 4 hours and align with the midnight local time (0000, 0400, 0800, 1200, 1600, and 2000):

```
ContentEngine(config)# transaction-logs archive interval every-day every 4
```

The following example shows how to export once a day at midnight local time:

```
ContentEngine(config)# transaction-logs export interval every-day every 24
```

The following example shows how to configure export intervals:

```
ContentEngine(config)# transaction-logs archive interval every-hour ?
  at        Specify the time at which to archive each day
  every     Specify interval in minutes. It will align with top of the hour

ContentEngine(config)# transaction-logs archive interval every-hour at ?
  <0-59>  Specify the minute alignment for the hourly archive
ContentEngine(config)# transaction-logs archive interval every-hour every ?
  <2-30>  Interval in minutes: {2, 5, 10, 15, 20, 30}
```

The following example shows how to create a separate log on the remote syslog host for real-time transaction log entries and how to configure a unique facility (for example, local1) on the Content Engine:

```
ContentEngine(config)# transaction-logs logging facility local1
```

Related Commands

clear transaction-log
show statistics transaction-logs
show transaction-logging
transaction-log force

tvout

To enable and configure TV-out service, use the **tvout** global configuration command. Use the **no** form of this command to disable TV-out service.

tvout {**enable** | **signal** {**ntsc** | **pal**}}

no tvout {**enable** | **signal** {**ntsc** | **pal**}}

Syntax Description

enable	Enables TV-out service.
signal	Configures the TV-out signal format.
ntsc	Sets the output signal to the National Television Systems Committee (NTSC) standard.
pal	Sets the output signal to the Phase Alternating Line (PAL) standard.

Defaults

The default TV-out signal format is NTSC.

Command Modes

global configuration

Usage Guidelines

The TV-out service provides capability for local playback of the pre-positioned MPEG content through a hardware decoder that converts the digital information into an analog TV signal. The TV-out service is enabled or disabled on the Content Engine. The **tvout enable** command is only functional if a supported MPEG hardware decoder is installed.

You can combine media from multiple channels in the playlist and add image overlays to media files in TV-out programs. Before you create a TV-out program, you must first enable the TV-out feature on the Content Engine and designate the format of the signal output.

Content Engines that are equipped with an integrated Moving Picture Experts Group (MPEG) decoder can convert digital media into analog TV signals. These Content Engines play media files using NTSC or PAL video signals and can play video directly to a TV monitor for applications such as kiosks, cable TV systems, and video walls.

Note The pre-positioned content is only supported on registered Content Engines; it is not supported on Content Engines that are not registered with a Content Distribution Manager and are being managed and monitored with the Content Engine GUI or CLI. The TV-out service, which involves the pre-positioned content, is only available on Content Engines that are registered with a Content Distribution Manager and not supported on standalone or locally deployed Content Engines.

In the ACNS 5.2.3 software and later releases, the following changes that are related to the TV-out service were incorporated:

- In the ACNS 5.2.3 software and later releases, the ACNS TV-out functionality works for the CE-510 and CE-565 models equipped with newer Vela II Revision D and Revision E MPEG hardware decoder cards. (In the ACNS 5.2 software, this functionality did not work for these cards.)
- New driver software was incorporated into the ACNS 5.2.3 software and later releases. This new driver software supports both the existing Vela II Revision A cards and the newer Vela II Revision D and Revision E cards.

Examples

The following example enables TV-out on the Content Engine:

```
ContentEngine(config)# tvout enable
```

The following example configures TV-out format on the Content Engine as NTSC:

```
ContentEngine(config)# tvout signal ntsc
```

The following example configures TV-out format on the Content Engine as PAL:

```
ContentEngine(config)# tvout signal pal
```

Related Commands

show running-config
show tvout

type

To display a file, use the **type** EXEC command.

type *filename*

Syntax Description

filename	Name of file.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to display the contents of a file within any Content Engine file directory. This command may be used to monitor features such as transaction logging or system logging (syslog).

Examples

The following example displays the syslog file on the Content Engine:

```
ContentEngine# type /local1/syslog.txt
Jan 10 22:02:46 (none) populate_ds: %CE-CLI-5-170050: Cisco ACNS software starts booting
Jan 10 22:02:47 (none) create_etc_hosts.sh: %CE-CLI-5-170051: HOSTPLUSDOMAIN: NO-HOSTNAME
Jan 10 22:02:47 NO-HOSTNAME : %CE-CLI-5-170053: Recreated etc_hosts (1, 0)
Jan 10 22:02:48 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [CLI_VER_NTP] requests stop
service ntpd
Jan 10 22:02:49 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [ver_tvout] requests stop
service tvoutsvr
Jan 10 22:02:50 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330084: [ver_rtspg] requests restart
service rtspg
Jan 10 22:02:50 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [ver_iptv] requests stop
service sbss
Jan 10 22:02:51 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330080: [ver_telnetd] requests start
service telnetd
Jan 10 22:02:52 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [ver_wmt] requests stop service
wmt_mms
Jan 10 22:02:53 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [ver_wmt] requests stop service
wmt_logd
Jan 10 22:02:55 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [Unknown] requests stop service
mcast_sender
Jan 10 22:02:55 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330082: [Unknown] requests stop service
mcast_receiver
Jan 10 22:02:56 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330024: Service 'populate_ds' exited
normally with code 0
Jan 10 22:02:56 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330040: Start service 'parser_server'
using: '/ruby/bin/parser_server' with pid: 1753
Jan 10 22:02:56 NO-HOSTNAME Nodemgr: %CE-NODEMGR-5-330040: Start service
'syslog_bootup_msgs' using: '/ruby/bin/syslog_bootup_msgs' with pid:
1754
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>Linux version 2.4.16
(cnbuild@builder2.cisco.com) (gcc version 3.0.4) #1
SMP Fri Jan 7 19:26:58 PST 2005
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <6>setup.c: handling
flash window at [15MB..16MB)
```

```
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <6>BIOS-provided
physical RAM map:
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
0000000000000000 - 000000000009ec00 (usable)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
000000000009ec00 - 00000000000a0000 (reserved)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
00000000000e0800 - 0000000000100000 (reserved)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
0000000000100000 - 0000000000f00000 (usable)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
0000000000f00000 - 0000000001000000 (reserved)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
0000000001000000 - 0000000010000000 (usable)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4> BIOS-e820:
00000000fff00000 - 0000000100000000 (reserved)
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <6>setup.c: reserved
bootmem for INITRD_START = 0x6000000, INITRD_SIZE = 117
09348
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>On node 0 totalpages:
65536
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>zone(0): 4096 pages.
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>zone(1): 61440 pages.
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>zone(2): 0 pages.
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>Local APIC disabled
by BIOS -- reenabling.
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>Found and enabled
local APIC!
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <4>Kernel command line:
root=/dev/ram ramdisk_size=100000 ramdisk_start=0x60
00000 console=ttyS0,9600n8
Jan 10 22:02:56 NO-HOSTNAME syslog_bootup_msgs: %CE-SYS-5-900001: <6>Initializing CPU#0
--More--
.
.
```

Related Commands

cpfile
dir
lls
ls
mkfile

type-tail

To view a specified number of lines of the end of a log file or to view the end of the file continuously as new lines are added to the file, use the **type-tail** command in EXEC mode.

type-tail *filename* [*line* | **follow**]

Syntax Description

filename	File to be examined.
line	(Optional) Number of lines from the end of the file to be displayed (1–65535).
follow	(Optional) Displays the end of the file continuously as new lines are added to the file.

Defaults

Ten lines shown

Command Modes

EXEC

Usage Guidelines

This command allows you to monitor a log file by letting you view the end of the file. You can specify the number of lines at the end of the file that you want to view, or you can follow the last line of the file as it continues to log new information. To stop the last line from continuously scrolling, press **Ctrl-C**.

Examples

The following example shows the list of log files in the /local1 directory:

```
stream-ce# ls /local1
WS441
Websense
WebsenseEnterprise
Websense_config_backup
WsInstallLog
badfile.txt
codecoverage
core.stunnel.5.3.0.b100.cnbuild.5381
core_dir
crash
crka.log
cse_live
cse_vod
dbdowngrade.log
dbupgrade.log
downgrade
errorlog
http_authmod.unstrip
index.html
logs
lost+found
netscape-401-proxy
netscape-401-proxy1
netscape-dump
newwebsense
oldWsInstallLog
preload_dir
proxy-basic1
```

```
proxy1
proxy2
proxy3
proxy4
proxy5
proxy6
proxy7
proxy8
proxyreply
proxyreply-407
real_vod
ruby.bin.cli_fix
ruby.bin.no_ws_fix
ruby.bin.ws_edir_fix
sa
service_logs
smartfilter
smfnaveen
superwebsense
syslog.txt
syslog.txt.1
syslog.txt.2
temp
two.txt
url.txt
urllist.txt
var
vpd.properties
websense.pre-200
webtarball44
webtarball520
wmt_vod
ws_upgrade.log
```

The following example displays the last ten lines of the syslog.txt file. In this example, the number of lines to display is not specified; however, ten lines is the default.

```
stream-ce# type-tail /local1/syslog.txt
Oct  8 21:49:15 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:15 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:15 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:17 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:17 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:17 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:19 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:19 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:19 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:21 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
```

The following example displays the last 20 lines of the syslog.text file:

```
stream-ce# type-tail /local1/syslog.txt 20
Oct  8 21:49:11 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:11 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:13 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
```

```
Oct  8 21:49:13 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:13 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:15 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:15 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:15 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:17 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:17 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:17 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:19 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:19 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:19 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:21 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:21 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:21 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:23 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:23 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:23 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
```

The following example follows the file as it grows:

```
stream-ce# type-tail /local1/syslog.txt ?
  <1-65535>  The numbers of lines from end
  follow     Follow the file as it grows
  <cr>
stream-ce# type-tail /local1/syslog.txt follow
Oct  8 21:49:39 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:41 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:41 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:41 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:43 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:43 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:43 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:45 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:45 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:45 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:47 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:47 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:47 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
Oct  8 21:49:49 stream-ce syslog:(26830)TRCE:input_serv.c:83-> select_with
return 0, ready = 0
Oct  8 21:49:49 stream-ce syslog:(26832)TRCE:al_master.c:246-> select_with
return 0, ready = 0
Oct  8 21:49:49 stream-ce syslog:(26832)TRCE:in_mms.c:1747-> tv = NULL
```

undebug

To disable debugging functions, use the **undebug** EXEC command.

undebug *option*

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

We recommend that you use the **debug** and **undebug** commands only at the direction of Cisco TAC. See the "Obtaining Technical Assistance" section on page -xvii for more information.

See the "debug" section on page 2-128 for more information about debug functions.

Related Commands

debug
no debug
show debug

url-filter (EXEC)

To reload new local good sites or good sites' lists on the Content Engine when the **url-filter** feature is enabled, use the **url-filter local-list-reload** EXEC command.

url-filter local-list-reload {**http** | **rtsp** | **wmt**}

Syntax Description

local-list-reload	Reloads the local lists of bad and good URLs when the **url-filter** global configuration command is enabled.
http	Specifies the local file lists for HTTP requests (HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests).
rtsp	Specifies the local file lists for requests over RTSP (requests from RealMedia clients).
wmt	Specifies the local file lists for WMT requests, and RTSP-over-RTP [the standard IETF RTSP protocol with Microsoft proprietary extensions] requests from Windows Media 9 players).

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

When you update the badurl.lst or goodurl.lst file, use the **url-filter local-list-reload** command to reload the latest good sites or good sites' lists that you created or edited using the **url-filter** command.

Examples

The following example shows how to reload new good sites' list or bad sites' list on the Content Engine for HTTP:

```
ContentEngine# url-filter local-list-reload http
```

The following example shows how to reload new good sites' list or bad sites' list on the Content Engine for RTSP:

```
ContentEngine# url-filter local-list-reload rtsp
```

The following example shows how to reload new good sites' list or bad sites' list on the Content Engine for WMT:

```
ContentEngine# url-filter local-list-reload wmt
```

Related Commands

url-filter (global configuration mode)

url-filter (global configuration)

To configure URL filtering over HTTP, RTSP, or WMT, use the **url-filter** global configuration command. Use the **no** form of this command to disable selected options.

url-filter http bad-sites-deny {**enable** | **file** *filename*}

url-filter http custom-message *dirname*

url-filter http good-sites-allow {**enable** | **file** *filename*}

url-filter http N2H2 {**allowmode enable** | **enable** | **server** {*hostname* | *ip-address*} [**port** *portnum* [**timeout** *seconds*]]}

url-filter http smartfilter enable

url-filter http websense {**allowmode enable** | **enable** | **server** {*hostname* | *ip-address* | **local**}[**port** *portnum* [**timeout** *seconds* [**connections** *connection*]]]}

url-filter rtsp {**bad-sites-deny** {**enable** | **file** *filename*}}

url-filter rtsp {**good-sites-allow** {**enable** | **file** *filename*}}

url-filter wmt {**bad-sites-deny** {**enable** | **file** *filename*}}

url-filter wmt {**good-sites-allow** {**enable** | **file** *filename*}}

no url-filter {**http** {**bad-sites-deny** {**enable** | **file**} | **custom-message** | **good-sites-allow** {**enable** | **file**} | **N2H2** {**allowmode enable** | **enable** | **server**} | **smartfilter enable** | **websense** {**allowmode enable** | **enable** | **server** {*hostname* | *ip-address* | **local**} [**port** *portnum* [**timeout** *seconds* [**connections** *connection*]]]}} | {**rtsp** | **wmt**}{**bad-sites-deny** {**enable** | **file**} | **good-sites-allow** {**enable** | **file**}}}

Syntax Description

http	Configures HTTP requests.
bad-sites-deny	Configures the local good sites' list for HTTP access denial.
enable	Enables URL filtering of the local good sites' list over HTTP.
file	Specifies the HTTP good sites' list filename.
filename	Name of the file that contains the HTTP good sites' list.
custom-message	Sends the customized access-denied message over HTTP. Specifies the directory that contains the block.html file.
dirname	Name of the directory that contains the block.html file and related graphics.
good-sites-allow	Configures the local good sites' list for allowing HTTP access.
enable	Enables URL filtering of the local good sites' list over HTTP.
file	Specifies the HTTP good sites' list filename.
filename	Name of the file that contains the HTTP good sites' list.
N2H2	Configures the HTTP N2H2 server to determine the URL access policy.
allowmode	Enables HTTP access to a site if the N2H2 server does not respond.
enable	Enables N2H2 allowmode over HTTP.
enable	Enables HTTP N2H2 filtering.

server	Configures HTTP N2H2 server parameters.
hostname	Hostname of the HTTP N2H2 server.
ip-address	IP address of the HTTP N2H2 server.
port	(Optional) Configures the HTTP N2H2 server port number.
portnum	Port on which to send the N2H2 requests (1–65535) over HTTP.
timeout	(Optional) Configures the maximum time to wait for a response from the N2H2 server over HTTP.
seconds	Timeout value in seconds (1–20). The default is 5 seconds.
smartfilter	Configures SmartFilter URL filtering over HTTP.
enable	Enables SmartFilter filtering over HTTP.
websense	Configures HTTP Websense parameters.
allowmode	Allows HTTP access to a site if Websense server does not respond.
enable	Enables Websense allow mode over HTTP.
enable	Enables Websense filtering over HTTP.
server	Specifies the HTTP Websense server.
hostname	Hostname of the external HTTP Websense server.
ip-address	IP address of the external HTTP Websense server.
local	Uses the local Websense server.
port	(Optional) Establishes the HTTP Websense server port number.
portnum	Port on which to send the Websense requests (1–65535) over HTTP.
timeout	(Optional) Configures the maximum time to wait for a response from Websense server over HTTP.
seconds	Timeout value in seconds (0–240).
connections	(Optional) Configures the number of persistent connections to Websense server. Do not change the default unless you know for certain that a different value is required.
connection	Number of persistent connections per CPU (default is 40 per CPU).
rtsp	Configures RTSP requests.
bad-sites-deny	Configures the RTSP good sites' list for access denial.
enable	Enables URL filtering of the local good sites' list over RTSP.
file	Specifies the RTSP good sites' list filename.
filename	Name of the file that contains the RTSP good sites' list.
good-sites-allow	Configures the good sites' list for allowing RTSP access.
enable	Enables URL filtering of the local good sites' list over RTSP.
file	Specifies the RTSP good sites' list filename.
filename	Name of the file that contains the RTSP good sites' list.
wmt	Configures WMT requests. **Note** In the ACNS 5.3 software release, the **url-filter wmt** command was modified to support local list URL filtering for RTSP requests from Windows Media 9 players.
bad-sites-deny	Configures the WMT good sites' list for access denial.
enable	Enables the good sites' list URL filtering over WMT.
file	Specifies the WMT good sites' list filename.

filename	Name of the file that contains the WMT good sites' list.
good-sites-allow	Configures the good sites' list for allowing access over WMT.
enable	Allows good sites' list URLs over WMT.
file	Specifies the WMT good sites' list filename.
filename	Name of the file that contains the WMT good sites' list.

Defaults

url-filter N2H2 allowmode: enabled

url-filter websense allowmode: enabled

N2H2 server port *portnum*: 4005

N2H2 sever timeout *seconds*: 5

websense server port *portnum*: 15868

websense server timeout *seconds*: 20

connections: 40 per CPU

Command Modes

global configuration

Usage Guidelines

Table 2-179 lists the various URL filtering schemes that you can use to configure a Content Engine to control client access to websites.

Table 2-179 URL Filtering Mechanisms with Content Engines

URL Filtering Scheme	More Information
Local list files	Deny access to URLs specified in a list. Permit access only to URLs specified in a list. See the "URL Filtering with Local Lists" section on page 2-825.
N2H2 external servers	Direct client requests for content to an external N2H2 server for URL filtering. See the "URL Filtering with the N2H2 Server" section on page 2-828.
Websense server	Direct client requests for content to either the local Websense plug-in or to an external Websense server for URL filtering. See the "URL Filtering with the Websense Enterprise Server" section on page 2-828.
SmartFilter plug-in	Direct client requests for content to the SmartFilter plug-in for URL filtering. See the "URL Filtering with SmartFilter Software" section on page 2-829.

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 sites' list and bad sites' list) 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.

URL filtering allows the Content Engine to control client access to websites in any of the following ways:

- Deny access to URLs specified in a list (HTTP and RSTP traffic).
- Permit access only to URLs specified in a list (HTTP and RSTP traffic).
- Direct traffic to an N2H2 server for filtering (HTTP traffic only).
- Direct traffic to a Websense enterprise server for filtering (HTTP, FTP over HTTP, HTTPS over HTTP traffic, and HTTP and HTTPS WCCP-intercepted requests only).
- Filter traffic with Secure Computing Corporation SmartFilter Software, Release 4.0 (HTTP traffic, HTTPS over HTTP traffic, and HTTP and HTTPS WCCP-intercepted requests only).

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), MMS is used instead of RTSP to service the content request from the Windows Media player.

Note 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. Bypass features, the **error-handling transparent** and **bypass load** global configuration commands, are enabled initially by default and must be disabled manually. For error handling, use the **error-handling send-cache-error** or **error-handling reset-connection** command instead.

URL filtering existed in the Cache software 2.x releases. URL filtering in the ACNS 4.x and 5.x software differs from the URL feature in other releases as follows:

- An **enable** command option now exists for the **good-sites-allow** and **bad-sites-deny** options.
- The URL list filenames and customized blocking message directory name are now specified in the CLI.
- The **url-filter local-list-reload** command now dynamically refreshes a local URL list.
- The option **bad-sites-block** has been changed to **bad-sites-deny**.
- The **websense-server enable** command has been added to enable the local Websense plug-in for URL filtering.

URL Filtering with Local Lists

You can configure 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 protocols) and streaming media protocols such as RTSP. 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. The HTTP local list file should contain only HTTP, HTTPS, or FTP URLs. The HTTP local list file should contain only the following types of URLs; HTTP, HTTPS, or FTP URLs. In the ACNS 5.3 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. We strongly recommend that you track the local list file size and ensure that it does not become so large that it degrades performance.

You can configure a 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 with RealNetworks proprietary extensions)
- WMT requests (RTSP-over-RTP for Windows Media 9 clients and Windows Media 9 servers)

Note Filtering for native FTP and native HTTPS requests is not supported.

Support for RTSP-over-RTP (referred to as WMT RTSP requests) for Windows Media 9 players is available in the ACNS 5.3 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 RTSPT 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 was added in the ACNS 5.3 software.

For WMT URL filtering, filtering for RTSP URLs (rtsp://) 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.

Local list URL filtering is the only supported filtering mechanism for WMT requests and RTSP requests (requests from RealMedia players). The third-party URL filtering mechanisms (N2H2, SmartFilter, and Websense software) are not supported for WMT and RTSP requests. For HTTP requests, the local list URL filtering mechanism and N2H2, SmartFilter, and Websense URL filtering are supported.

URL Filtering with URL Lists

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

Note Replace HTTP with either RTSP or WMT in the examples that follow if you want to use URL lists on these protocols. Make sure the URLs that are either accessed or denied match the protocol used in the command.

Use the **url-filter wmt** global configuration commands to configure local list URL filtering for WMT requests. If you configure an RTSP URL in a WMT good sites' list, it blocks both the RTSPT and RTSPU URLs and the RTSP URL that is specified in the good sites' list.

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

Custom Blocking Messages

In the case of local list URL filtering, the Content Engine with the ACNS 5.x software can be configured to return a customized blocking message to the client. The custom message must be an administrator-created HTML page named block.html. Make sure to copy all embedded graphics associated with the custom message HTML page to the same directory that contains the block.html file. To enable the customized blocking message, use the **url-filter http custom-message** command and specify the directory name.

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

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

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

In the block.html file, objects (such as .gif, .jpeg, and so on) must be referenced with the string /content/engine/blocking/url, as shown as follows:

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

A 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
```

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

RADIUS and URL Filtering

When both RADIUS authentication and URL filtering are enabled on the Content Engine, users can be configured to bypass URL filtering using the user Filter-Id attribute in the RADIUS server database.

The following is an example of a user Filter-Id attribute entry in the RADIUS server database:

```
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. If blocking is not specified, Yes-Web-Blocking is the default RADIUS filter. Yes-Web-Blocking means that the request is subject to URL filtering and No-Web-Blocking means that the request is not subject to the URL filtering.

URL Filtering with the N2H2 Server

Content Engines can use an N2H2 enterprise server as a filtering engine and enforce the filtering policy configured on the N2H2 server. The 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.

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 mechanism is applied, whether the requested object is in the cache or not.

URL Filtering with the Websense Enterprise Server

The Content Engine can use a Websense enterprise server as a filtering engine and enforce the filtering and enforce the filtering policy configured on Websense server. You can also configure a 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 software, Release 5.1.x and earlier releases, only one Websense server is supported. In the ACNS software, Release 5.2 and later releases, up to two Websense servers are supported.

About Websense Server Failover

In the ACNS software, Release 5.2, a Websense server failover feature was added. 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 2-180 lists the supported Websense server failover configurations.

Table 2-180 Supported Websense Server Failover Configurations

Supported Configurations	Local (Internal) Websense Server	Remote Websense Server
Option A	The local Websense server is disabled on the Content Engine.	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).
Option B	The local Websense server is acting as the primary Websense server.	The secondary Websense server is running on an external host.
Option C	The local Websense server is acting as the secondary Websense server.	The primary Websense server is running on an external host.

The order in which you configure Websense servers determines which server is designated as the primary Websense server. The first configured Websense server is designated as the primary server. Configuration of a secondary Websense server is optional.

If both the primary and secondary Websense servers fail (if there is no response from Websense server), the Content Engine will time out all HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) and either allow or block all traffic based on the **allowmode** configuration.

If the primary Websense server is down, HTTP requests are redirected to the configured secondary Websense server. The Content Engine sends a request to the primary Websense server at certain polling intervals to determine when the primary Websense server comes back online. If the secondary Websense server is also down (assuming that the primary Websense server is already down), the Content Engine sends a request to the secondary Websense server at certain polling intervals to determine when the secondary Websense server comes back online. All HTTP requests received at that time are either blocked or allowed depending on the allowmode configuration. When the designated primary Websense server is back online, the Content Engine establishes connections with the primary server and sends client requests for filtering.

Also, primary and secondary Websense servers can be running any possible combination of Websense server Version 4.0.3 or later. While downgrading from the ACNS 5.2 and later releases to an earlier release of the ACNS 5.x software, the secondary Websense server configuration, if any, would be removed.

URL Filtering with SmartFilter Software

SmartFilter software running on Content Engines provides employee Internet management (EIM) functionality when used with proxy servers, firewalls, and caching appliances. SmartFilter filtering 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 Content Engine uses a suite of plug-in APIs to allow the SmartFilter software to implement hooks at strategic points during an HTTP transaction.

Note 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 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.

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 SmartFilter 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. Use the **url-filter http smartfilter enable** global configuration command to enable SmartFilter URL filtering on a Content Engine**.** To use SmartFilter URL filtering with a cluster of 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 5.4 software supports SmartFilter software Version 4.0.1.

About the SmartFilter Control List

These 30 categories are set to Deny in the default SmartFilter software policy. SmartFilter software also provides ten user-defined categories that allow you to define and filter sites that are not included in the SmartFilter Control List. 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.

Note For more information about configuring the SmartFilter software, go to the following website: http://www.securecomputing.com.

Examples

The following example shows how to block a list of URLs:

```
ContentEngine(config)# url-filter http bad-sites-deny badurl.lst
```

The following example shows how to disable URL blocking:

```
ContentEngine(config)# no url-filter http bad-sites-deny enable
ContentEngine(config)# no url-filter http good-sites-allow enable
```

The following example shows how to enable a custom message by specifying the directory in which the block.html file is located and then entering the **enable** command:

```
ContentEngine(config)# url-filter http custom-message /local1/url_dir
ContentEngine(config)# url-filter http custom-message enable
```

The following example shows how to configure a Content Engine to use N2H2 URL filtering with IP address 172.16.22.10, port 4008, and a 6-second timeout:

```
ContentEngine(config)# url-filter http N2H2 server 172.16.22.10 port 4008 timeout 6
```

The following example shows how to configure a Content Engine to use Websense URL filtering with IP address 172.16.11.22, port 15900, and a 4-second timeout:

```
ContentEngine(config)# url-filter websense server 172.16.11.22 port 15900 timeout 4
```

The following example shows how to enable a Content Engine to use SmartFilter URL filtering:

```
ContentEngine(config)# url-filter smartfilter enable
starting smartfilter
```

The following example shows how to display a Websense server configuration:

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

Local list configurations
=================================
Good-list file name : /local1/good.list
Bad-list file name : /local1/bad.list
Custom message directory :

Websense server configuration
=================================
Websense server IP     : <none>
Websense server port   : 15868
Websense server timeout: 20 (in seconds)
Websense server connections: 40
Websense allow mode is ENABLED

N2H2 server configuration
=============================
N2H2 server IP       : <none>
N2H2 server port     : 4005
N2H2 server timeout  : 5 (in seconds)
N2H2 allow mode is ENABLED
```

Related Commands

clear statistics url-filter http
debug url-filter
show statistics url-filter http
show url-filter
url-filter local-list-reload (EXEC mode)
websense-server enable

username

To establish username authentication, use the **username** global configuration command.

username *name* {{**cifs-password** | **samba-password**} {**0** *plainword* | **1** *lancrypto ntcrypto* | *cleartext*} | **password** {**0** *plainword* | **1** *cryptoword* | *cleartext*} [**uid** *uid*] | **privilege** {**0** | **15**}}

no username *name*

Syntax Description

name	Username.
cifs-password	Sets the Windows user password.
samba-password	Deprecated, same as **cifs-password**.
0	Specifies a clear-text password. This is the default password setting.
plainword	Clear-text user password.
1	Specifies a type 1 encrypted password.
lancrypto	Encrypted password for LAN Manager networks.
ntcrypto	Encrypted password for Windows NT networks.
cleartext	Unencrypted (clear-text) password for Windows NT networks.
password	Sets the user password.
cryptoword	Encrypted user password.
uid	Sets the user ID for a clear-text password or an encrypted password.
uid	Encrypted password user ID (2001–65535).
privilege	Sets the user privilege level.
0	Sets the user privilege level for a normal user.
15	Sets the user privilege level for a superuser.

Defaults

The **password** value is set to 0 (clear text) by default.

Default administrator account:

- **Uid**: 0
- **Username**: admin
- **Password**: default
- **Privilege**: superuser (15)

Command Modes

global configuration

Usage Guidelines

The **username** global configuration command changes the password and privilege level for existing user accounts.

User Authentication

User access is controlled at the authentication level. For every HTTP or HTTPS request that applies to the administrative interface, including every CLI and API request that arrives at the ACNS network devices, the authentication level has visibility into the supplied username and password. Based on CLI-configured parameters, a decision is then made to either accept or reject the request. This decision is made either by checking local authentication or by performing a query against a remote authentication server. The authentication level is decoupled from the authorization level, and there is no concept of role or domain at the authentication level.

When local CLI authentication is used, all configured users can be displayed by entering the **show running-config** command. Normally, only administrative users need to have username authentication configured.

Note Every ACNS network device should have an administrative password that can override the default password.

User Authorization

Domains and roles are applied by the Content Distribution Manager at the authorization level. Requests must be accepted by the authentication level before they are considered by the authorization level. The authorization level regulates the access to resources based on the Content Distribution Manager GUI role and domain configuration.

Regardless of the authentication mechanism, all user authorization configuration is visible in the GUI.

Examples

When you first connect an ACNS device to an ACNS network, you should immediately change the password for the username *admin*, which has the password *default*, and the privilege level superuser.

The following example shows how to change the password:

```
ContentEngine(config)# username admin password yoursecret
```

The following example shows how passwords and privilege levels are reconfigured:

```
ContentEngine# show user username abeddoe
Uid                 : 2003
Username            : abeddoe
Password            : ghQ.GyGhP96K6
Privilege           : normal user
ContentEngine# show user username bwhidney
Uid                 : 2002
Username            : bwhidney
Password            : bhlohlbIwAMOk
Privilege           : normal user
ContentEngine(config)# username bwhidney password 1 victoria
ContentEngine(config)# username abeddoe privilege 15
User's privilege changed to super user (=15)
ContentEngine# show user username abeddoe
Uid                 : 2003
Username            : abeddoe
Password            : ghQ.GyGhP96K6
Privilege           : super user

ContentEngine# show user username bwhidney
Uid                 : 2002
Username            : bwhidney
Password            : mhYWYw.7P1Ld6
Privilege           : normal user
```

Related Commands **show user**
show users

wccp access-list

To configure an IP access list for inbound WCCP GRE-encapsulated traffic, use the **wccp access-list** global configuration command.

wccp access-list {*acl-name* | *acl-num*}

no wccp access-list [*acl-name* | *acl-num*]

Syntax Description

acl-name	Alphanumeric identifier of up to 30 characters, beginning with a letter that identifies the ACL to apply to the current interface.
acl-num	Numeric identifier that identifies the access list to apply to the current interface. For standard access lists, the valid range is 1–99; for extended access lists, the valid range is 100–199.

Defaults

WCCP access lists are not configured by default.

Command Modes

global configuration

Usage Guidelines

In the ACNS 5.2 software and later releases, you can 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. Because the default is that no WCCP access list is configured, the WCCP access lists are not shown as part of the Content Engine's configuration.

You can enable WCCP applications to be attached to a particular interface (such as management services to the private IP address space) so that the Content Engine can have one interface in the customer's IP address space that serves the content, and another interface in a private IP address space that the administrator uses for management purposes. This setting ensures that clients can access the Content Engine only in the public IP address space for serving the content and not access it for management purposes. A device attempting to access one of these applications associated with an access control list (ACL) must be on the list of trusted devices to be allowed access.

The **wccp access-list** *acl-num* global configuration command configures an access control list to allow access to WCCP applications. The *acl-num* variable is a number in the range 1 to 99, indicating a standard access control list or a number in the range 100 to 199, indicating an extended access control list. WCCP checks against the specified access control list before accepting or dropping incoming packets.

See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to use standard IP ACLs to control WCCP access on a Content Engine.

Examples

The following example configures the Content Engine to apply IP access list number 10 to the inbound WCCP traffic:

```
ContentEngine(config)# wccp access-list 10
```

The following example shows sample output from the **show ip access-list** EXEC command from a Content Engine that has several WCCP access lists configured:

```
ContentEngine# show ip access-list
Space available:
    40 access lists
   489 access list conditions

Standard IP access list 10
   1 deny 10.1.1.1
   2 deny any
     (implicit deny any: 0 matches)
   total invocations: 0
Standard IP access list 98
   1 permit any
     (implicit deny any: 0 matches)
   total invocations: 0
Extended IP access list 100
   1 permit icmp any any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list 101
   1 permit ip any any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list 102
   1 permit icmp 0.0.1.1 255.255.0.0 any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list 111
   1 permit gre 0.1.1.1 255.0.0.0 any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list 112
   1 permit ip any any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list 113
   1 permit gre 0.1.1.1 255.0.0.0 any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list ext_acl_2
   1 permit gre any any
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0
Extended IP access list extended_ip_acl
   1 permit tcp any eq 2 any eq exec
     (implicit fragment permit: 0 matches)
     (implicit deny ip any any: 0 matches)
   total invocations: 0

Interface access list references:
  PortChannel     2    inbound   extended_ip_acl
  PortChannel     2    outbound  101

Application access list references:
  snmp-server                    standard  2
```

```
    UDP ports: none (List Not Defined)
  tftp_server                     standard  98
    UDP ports:    69
  WCCP                            either    10
    Any IP Protocol
ContentEngine#
```

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
Packets dropped due to loopback redirect:   0
Connections bypassed due to load:           0
Packets sent back to router:                0
Packets sent to another CE:                 0
GRE fragments redirected:                   0
Packets failed GRE encapsulation:           0
Packets dropped due to invalid fwd method:  0
Packets dropped due to insufficient memory: 0
Packets bypassed, no conn at all:           0
Packets bypassed, no pending connection:    0
Packets due to clean wccp shutdown:         0
Packets bypassed due to bypass-list lookup: 0
Packets received with client IP addresses:  0
Conditionally Accepted connections:         0
Conditionally Bypassed connections:         0
L2 Bypass packets destined for loopback:    0
Packets w/WCCP GRE received too small:      0
Packets dropped due to IP access-list deny: 29
L2 Packets fragmented for bypass:           0
Content Engine#
```

Related Commands

show ip access-list
show wccp gre

wccp custom-web-cache

To enable the Content Engine to accept redirected HTTP traffic on a port other than 80, use the **wccp custom-web-cache** global configuration command. To disable custom web caching, 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*]}

no wccp custom-web-cache

Syntax Description

mask	Sets the mask used for Content Engine assignment. Configure at least one mask; the maximum is four masks.
dst-ip-mask	(Optional) Sets the mask used to match the packet destination IP address.
hex_num	IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000–FE000000.
dst-port-mask	(Optional) Sets the mask used to match the packet destination port number.
port_hex_num	Port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0–65535.
src-ip-mask	(Optional) Sets the mask used to match the packet source IP address.
src-port-mask	(Optional) Sets the mask used to match the packet source port number.
router-list-num	Sets the router list number.
num	Router list number (1–8).
port	Sets the port number.
port	Port number (1–65535).
assign-method-strict	(Optional) Forces WCCP to strictly use only the configured assignment method. For more information, see the "Assignment Method" section on page 2-840.
hash-destination-ip	(Optional) Defines the load-balancing hash of the destination IP address (the default).
hash-destination-port	(Optional) Defines the load-balancing hash of the destination port.
hash-source-ip	(Optional) Defines the load-balancing hash of the source IP address.
hash-source-port	(Optional) Defines the load-balancing hash of the source port.
l2-redirect	(Optional) Sets the packet forwarding by Layer 2 redirect. For more information, see the "WCCP Layer 2 Support" section on page 2-839.
mask-assign	(Optional) Uses the mask method for the Content Engine assignment.
password	(Optional) Sets the authentication password to be used for secure traffic among the Content Engines within a cluster and the router for a specified service. **Note** Be sure to enable all other Content Engines and routers within the cluster with the same password.
key	WCCP service password key. Passwords must not exceed eight characters.

weight	(Optional) Sets the weight percentage for load balancing. For more information, see the "Load Balancing" section on page 2-841.
percentage	Percentage value (0–100).

Defaults

wccp custom-web-cache: default

dst-ip-mask: 0x00001741

src-ip-mask: 0x00000000

dst-port-mask: 0x0

src-port-mask: 0x0

Command Modes

global configuration

Usage Guidelines

The **wccp custom-web-cache** 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, service 98 must be enabled on the router. WCCP Version 1 does not support custom web caching.

Transparent caching on ports other than port 80 can be performed by the Content Engine when WCCP is not enabled or when client browsers have previously been configured to use a legacy proxy server. See the **http proxy** command for further information.

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 the hardware at Layer 2. This WCCP function 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 more efficient 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 by entering the **l2-redirect** option in the CLI.

Layer 2 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. 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 sends 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.

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. Because the upper five bits of the IP multicast address are dropped in this mapping, the resulting address is not unique; 32 different multicast group IDs map to the same Ethernet address. 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.

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, and 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 method is not supported through the Content Engine GUI).

Note For information on the default hashing assignment for WCCP Version 2 services, see Chapter 4 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

Note The default hashing assignment for predefined WCCP services is fixed and cannot be changed.

Assignment Method

The assignment method denotes the method used by WCCP to perform load distribution across Content Engines. There are two possible load-balancing methods assignment methods: hashing and masking. If the mask load-balancing method is not specified, then the hash load-balancing method, 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 mask assignment is used to refer to WCCP Layer 2 Policy Feature Card 2 (PFC2) input redirection.

The Content Engine falls back to the assignment method supported in the hardware unless the **assign-method-strict** option is used (for example, if the **wccp custom-web-cache assign-method-strict** command is used to specify the **assign-method-strict** option for the custom-web-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 packets have not returned. It does not redirect the packets that have been returned, because the Content Engine has determined that these 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.

Load Balancing

WCCP Version 2 supports dynamic load distribution that allows the routers to adjust the loads that are forwarded to the individual Content Engines in the cluster. It uses two techniques to perform this task:

The **weight** parameter represents a percentage of the load that is 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.

See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure the custom-web-cache service (service 98) on a Content Engine and a router.

Examples

The following example shows the configuration for starting custom web caching on Ethernet interface 3 of a WCCP Version 2-enabled router:

```
Router(config): ip wccp 98

[Output not shown]

Router(config): ip interface ethernet 3
Router(config-if): ip web-cache 98 redirect out

[Output not shown]
```

The following example shows how to enable WCCP Version 2 on the 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 other than the standard web-cache service (service 0).

The following example creates a router list that specifies the routers that will support the custom-web-cache service. In this example, 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). The **l2-redirect** option specifies Layer 2 redirection as the packet-forwarding method (instead of GRE) and the **mask-assign** option specifies the mask assignment as the load-balancing method for this WCCP service.

```
ContentEngine(config)# wccp router-list 1 10.0.1.1 l2-redirect mask-assign
```

The following example shows how to inform the WCCP-enabled router in the specified router list that this Content Engine is accepting redirected custom web cache requests on port 31:

```
ContentEngine(config)# wccp custom-web-cache router-list-num 1 port 31
```

The following example shows how to turn on WCCP Version 2 on the router:

```
Router# configure terminal
Router(config)# ip wccp version 2
```

The following example shows the configuration on the Content Engine:

```
ContentEngine(config)# wccp custom-web-cache router-list-num 5 port 82 weight 30 password
Allied hash-destination-ip hash-source-port
ContentEngine(config)# http proxy outgoing ans.allied.com 82 no-local-domain
```

The following example shows the running configuration on the Content Engine:

```
ContentEngine# show running-config
Building configuration...
```

```
Current configuration:
!
....
!
http proxy outgoing 192.168.200.68 82 no-local-domain
!
wccp router-list 5 10.1.1.1
wccp custom-web-cache router-list 5 port 82 weight 30 password Allied hash-destination-ip
hash-source-port
wccp home-router 10.1.1.2
wccp version 2
!
end
```

Related Commands

http proxy incoming
http proxy outgoing
show wccp content-engines
show wccp flows web-cache
show wccp masks web-cache
show wccp routers
show wccp slowstart web-cache
show wccp status
wccp version 2
wccp web-cache

wccp dns

To allow a WCCP-enabled router to intercept packets, process them, and reinsert them into the request stream, use the **wccp dns** global configuration command. To disable this function, use the **no** form of this command.

wccp dns {**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**] [**hash-destination-ip**] [**hash-destination-port**] [**hash-source-ip**] [**hash-source-port**] [**l2-redirect**] [**mask-assign**] [**password** *key*] [**weight** *percentage*]}

no wccp dns

Syntax Description

mask	Sets the mask used for the Content Engine assignment. Configure at least one mask; the maximum is four masks.
dst-ip-mask	(Optional) Sets the mask used to match the packet destination IP address.
hex_num	IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000–FE000000.
dst-port-mask	(Optional) Sets the mask used to match the packet destination port number.
port_hex_num	Port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0–65535.
src-ip-mask	(Optional) Sets the mask used to match the packet source IP address.
src-port-mask	(Optional) Sets the mask used to match the packet source port number.
router-list-num	Sets the router list number.
num	Router list number (1–8).
assign-method-strict	(Optional) Forces WCCP to strictly use only the configured assignment method. For more information, see the "Assignment Method" section on page 2-840.
hash-destination-ip	(Optional) Defines the load-balancing hash of the destination IP address (the default).
hash-destination-port	(Optional) Defines the load-balancing hash of the destination port.
hash-source-ip	(Optional) Defines the load-balancing hash of the source IP address.
hash-source-port	(Optional) Defines the load-balancing hash of the source port.
l2-redirect	(Optional) Sets the packet forwarding by Layer 2 redirect. For more information, see the "WCCP Layer 2 Support" section on page 2-839.
mask-assign	(Optional) Uses the mask method for the Content Engine assignment.
password	(Optional) Sets the authentication password to be used for secure traffic among the Content Engines within a cluster and the router for a specified service. **Note** Be sure to enable all other Content Engines and routers within the cluster with the same password.
key	WCCP service password key. Passwords must not exceed eight characters.
weight	(Optional) Sets the weight percentage for load balancing. For more information, see the "Load Balancing" section on page 2-841.
percentage	Percentage value (0–100).

Defaults

wccp dns-cache: disabled

dst-ip-mask: 0x00001741

src-ip-mask: 0x00000000

dst-port-mask: 0x0

src-port-mask: 0x0

Command Modes

global configuration

Usage Guidelines

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 deregisters 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)."

The advantages of having the Content Engines intercept DNS requests are as follows:

- You can control the servers used through WCCP redirect lists, through bypass entries, or by having the Content Engine intercept requests to servers not in use and then reissuing requests to other servers.
- You can easily update and change the DNS servers as needed.
- Performance is improved (faster response time and lower bandwidth consumption) because of the DNS cache hits.

You can partition your network with a series of forwarding servers, each with its own DNS cache with the following results:

- You can effectively control DNS traffic across a WAN.
- You are better equipped to compensate for the loss of one or more servers, because the Content Engine can issue content requests to a different upstream server.

Typically, this partitioning also results in a performance improvement if a cluster of requests occurs, especially on high-latency interconnections.

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 as follows:

- 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.)

- Original DNS server—The DNS server from the original request (referred to as the original DNS server in this publication).

If DNS WCCP interception 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 Content Engine should use to resolve a domain name, as described in Table 2-181.

Table 2-181 Specifying DNS Servers for the DNS Caching Service with WCCP Version 2 Interception

CLI Command (Abbreviated Syntax)	Purpose
dns use-original-server only	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.
dns use-original-server after-configured	Configures the DNS cache service on a Content Engine to try the configured DNS servers first. If the configured DNS servers fail, the Content Engine tries the original DNS server.
dns use-original-server before-configured	Configures the DNS cache service on a Content Engine to try the original DNS server first and then the configured DNS servers.
no dns use-original-server	Configures the DNS cache service on a Content Engine to use only the list of configured DNS servers. This configuration is the default.

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.

See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure DNS caching for a Content Engine.

Examples

The following example shows the configuration for starting DNS caching on Ethernet interface 2 of a WCCP Version 2-enabled router:

Note WCCP Version 1 does not support the DNS caching service.

```
Router# configure terminal
Router(config)# ip wccp version 2
Router(config)# ip wccp 53
Router(config)# interface ethernet 2
Router(config-if)# ip wccp 53 redirect out
```

The following example configures the Content Engine to run WCCP Version 2 and enable the DNS caching service:

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

The following example shows how to 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 and then DNS caching is enabled on the Content Engine:

```
ContentEngine(config)# dns listen 10.1.1.0 port 53 hostname acme
ContentEngine(config)# dns enable
```

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, to redirect DNS requests to this Content Engine:

```
ContentEngine(config)# wccp router-list 1 10.77.157.41
```

The following example shows how to configure the Content Engine to accept redirected DNS requests for routers that are part of router list 1. The **l2-redirect** option specifies Layer 2 redirection as the packet-forwarding method (instead of GRE). In the following example, the **mask-assign** option specifies mask assignment as the load-balancing method for this WCCP service:

```
ContentEngine(config)# wccp dns router-list-num 1 l2-redirect mask-assign
```

Related Commands

dns
show dns
show statistics dns-cache
show wccp content-engines
show wccp flows dns
show wccp masks dns
show wccp routers
show wccp slowstart dns
show wccp status
wccp version 2

wccp flow-redirect enable

To enable WCCP flow redirection, use the **wccp flow-redirect enable** global configuration command. To disable flow redirection, use the **no** form of this command.

wccp flow-redirect enable

no wccp flow-redirect enable

Syntax Description

enable	Enables flow redirection.

Defaults

Enabled

Command Modes

global configuration

Usage Guidelines

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. 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.

Use the **wccp flow-redirect enable** global configuration command to implement WCCP flow protection. 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 allows the Content Engines to take a load that is appropriate for their capacity.

Note When bypass is enabled, the client tries to reach the origin web server. You must disable all bypass options to eliminate an unnecessary burden on the network.

Examples

The following example shows how to enable WCCP flow protection on a Content Engine:

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

The following example shows how to disable WCCP flow protection on a Content Engine:

```
ContentEngine(config)# no wccp flow-redirect enable
```

Related Commands

wccp slow-start enable

wccp ftp-native

To instruct the router to enable or disable transparent interception of FTP native traffic with WCCP Version 2, use the **wccp ftp-native** global configuration command. To disable this function, use the **no** form of this command.

wccp ftp-native {**mask** {**dst-ip-mask** *hex_num* [**src-ip-mask** *hex_num*] | **src-ip-mask** *hex_num* [**dst-ip-mask** *hex_num*]} | **router-list-num** *num* [**assign-method-strict**] [**l2-redirect**] [**mask-assign**] [**password** *key*] [**weight** *percentage*]}

no wccp ftp-native

Syntax Description

mask	Sets the mask used for the Content Engine assignment. Configure the mask for the destination IP address, the source IP address, or both.
dst-ip-mask	(Optional) Sets the mask used to match the packet destination IP address. The default is 0x00001741.
hex_num	IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000–FE000000.
src-ip-mask	(Optional) Sets the mask used in the packet source IP address. The default is 0x00000000.
router-list-num	Sets the router list number.
num	Router list number (1–8).
assign-method-strict	(Optional) Forces WCCP to strictly use only the configured assignment method. For more information, see the "Assignment Method" section on page 2-840.
l2-redirect	(Optional) Sets the packet forwarding by Layer 2 redirect. For more information, see the "WCCP Layer 2 Support" section on page 2-839.
mask-assign	(Optional) Uses the mask method for the Content Engine assignment.
password	(Optional) Sets the authentication password to be used for secure traffic among the Content Engines within a cluster and the router for a specified service. Be sure to enable all other Content Engines and routers within the cluster with the same password.
key	WCCP service password key. Passwords must not exceed eight characters.
weight	(Optional) Sets the weight percentage for load balancing. For more information, see the "Load Balancing" section on page 2-841.
percentage	Percentage value (0–100).

Defaults

wccp ftp-native: disabled

dst-ip-mask: 0x00001741

src-ip-mask: 0x00000000

Command Modes

global configuration

Usage Guidelines

Use this command to enable traffic interception of FTP native traffic with WCCP Version 2. With FTP native service, the router balances the traffic load within a Content Engine cluster based on the destination IP address (for example, FTP server IP address).

Note Transparent redirection of FTP requests is supported only by WCCP Version 2; transparent redirection through a Layer 4 switch is not supported.

You must set the **wccp router-list** command before you use this command.

Both **weight** and **password** are optional and can be used together or separately.

To enable the use of a password for secure FTP native traffic, use the **password** *key* option and be sure to enable all other Content Engines and routers within the cluster with the same password.

The **l2-redirect** option permits the Content Engine to receive transparently redirected FTP native traffic from a WCCP Version 2-enabled switch or router 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 the total load redirected to the Content Engine (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 a 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.

WCCP service ID 60 is associated with the FTP native service on the router, and it must be configured on the router using the **ip wccp 60** router configuration command.

The **wccp ftp-native router-list-number** and **wccp ftp-native mask** global configuration commands support native FTP caching on a Content Engine that is operating in transparent proxy mode.

See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure transparent FTP native caching with a Content Engine (transparent proxy server) and a WCCP Version 2-enabled router.

Examples

The following example configures WCCP to use the router list numbered 1:

```
ContentEngine(config)# wccp ftp-native router-list-num 1
```

The following example shows how to configure native FTP caching on a WCCP Version 2 router to transparently intercept FTP native requests and redirect them to the Content Engine by turning on native FTP caching. The service group number for this service is 60.

```
ContentEngine(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.

The following example shows how to specify an interface on which the native FTP caching service will run:

```
ContentEngine(config)# interface ethernet 0
```

The following example shows how to specify out for the native FTP caching service:

```
ContentEngine(config-if)# ip wccp 60 redirect out
```

The following example disables transparent interception of FTP traffic with WCCP Version 2:

```
ContentEngine(config)# no wccp ftp-native
```

The following example shows how to configure transparent FTP native caching on the Content Engine by defining a list of routers that will be used to redirect FTP native requests to this Content Engine:

```
ContentEngine(config)# wccp router-list 1 10.77.157.41
```

The 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.

The following example shows how to specify the router list that the Content Engine should accept redirected FTP native requests from:

```
ContentEngine(config)# wccp ftp-native router-list-num 1
```

The example shows how to configure the Content Engine to accept redirected FTP native requests for routers that are part of router list 1.

The following example shows how to specify Layer 2 redirection as the packet-forwarding method (instead of GRE) and specify the mask assignment as the load-balancing method (instead of 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)#
```

The **l2-redirect** option in the previous example specifies Layer 2 redirection as the packet-forwarding method (instead of GRE).

The following example shows how to 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.

The following example shows how to specify the maximum size of an FTP object for FTP native caching:

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

This parameter can be configured for either directory listings or particular objects in the cache. The previous example sets the maximum size for an FTP object size to 2 megabytes for FTP native caching.

The following example shows the output of the **show wccp modules** EXEC command to verify that the FTP proxy is enabled on the Content Engine:

```
ContentEngine# show wccp modules
Modules registered with WCCP on this Content Engine

Module  Socket  Expire(sec)  Name            Supported Services
------  ------  -----------  --------------- ------------------
2       13      4            WMT Proxy       WMT

3       14      4            MMSU WMT Proxy  MMSU

6       15      4            WMT-RTSPU       RTSPU

1       16      4            RTSP Proxy      RTSP

0       17      3            HTTP Proxy      Web Cache
```

```
                                                  Reverse Proxy
                                                  Custom Web Cache
                                                  HTTPS Cache
                                                  WCCPv2 Service 90
                                                  WCCPv2 Service 91
                                                  WCCPv2 Service 92
                                                  WCCPv2 Service 93
                                                  WCCPv2 Service 94
                                                  WCCPv2 Service 95
                                                  WCCPv2 Service 96
                                                  WCCPv2 Service 97

5       22      3            FTP Proxy       FTP
ContentEngine#
```

The following example shows the output of the **show wccp services** EXEC command to 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#
```

The following example shows the output of the **show ftp-native** EXEC command to view the current FTP native proxy configuration:

```
ContentEngine# show ftp-native
WCCP FTP service status:                 ENABLED
Maximum size of a FTP cacheable object:  2096128 KBytes
FTP data connection mode with Server:    Adhere to Client's mode (active or passive)
Incoming Proxy-Mode:
  Configured Proxy mode Native FTP connections on ports: 1 2 3 4 5 6 7 8
```

Related Commands

ftp-native
show ftp-native
show statistics ftp-native
show wccp content-engines
show wccp flows ftp
show wccp masks ftp
show wccp routers
show wccp slowstart ftp
show wccp status
wccp version 2

wccp home-router

To configure a WCCP Version 1 router IP address, use the **wccp home-router** global configuration command. To disable this function, use the **no** form of this command.

wccp home-router *ip-address*

no wccp home-router

Syntax Description

ip-address	Home router IP address.

Defaults

Disabled

Command Modes

global configuration

Usage Guidelines

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.

To use WCCP Version 1 with the Content Engine, you must also point the Content Engine to a designated home router by entering the **wccp home-router** *ip-address* command. This IP address may also be the address of the IP default gateway.

Make sure that WCCP Version 1 is enabled on the router.

Examples

The following example shows how to configure a WCCP Version 1 router IP address on the Content Engine:

```
ContentEngine(config)# wccp home-router 172.16.65.243
```

The following example shows how to disable the home router configured on the Content Engine:

```
ContentEngine(config)# no wccp home-router 172.16.65.243
```

Related Commands

show wccp routers
wccp version 1

wccp https-cache

To enable WCCP flow redirection to a Content Engine configured as an HTTPS server, use the **wccp https-cache** global configuration command. To disable this function, use the **no** form of this command.

wccp https-cache {**accept-all** | **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**] [**hash-destination-ip**] [**hash-destination-port**] [**hash-source-ip**] [**hash-source-port**] [**l2-redirect**] [**mask-assign**] [**password** *key*] [**weight** *percentage*]}

no wccp https-cache

Syntax Description

accept-all	Enables the the Content Engine to accept all HTTPS traffic by default, regardless of whether the origin HTTPS server is configured on the Content Engine.
mask	Sets the mask used for Content Engine assignment. Configure at least one mask; the maximum is four masks.
dst-ip-mask	(Optional) Sets the mask used to match the packet destination IP address.
hex_num	IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000–FE000000.
dst-port-mask	(Optional) Sets the mask used to match the packet destination port number.
port_hex_num	Source port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0–65535.
src-ip-mask	(Optional) Sets the mask used to match the packet source IP address.
src-port-mask	(Optional) Sets the mask used to match the packet source port number.
router-list-num	Sets the router list number.
num	Router list number (1–8).
assign-method-strict	(Optional) Forces WCCP to strictly use only the configured assignment method. For more information, see the "Assignment Method" section on page 2-840.
hash-destination-ip	(Optional) Defines the load-balancing hash of the destination IP address (the default).
hash-destination-port	(Optional) Defines the load-balancing hash of the destination port.
hash-source-ip	(Optional) Defines the load-balancing hash of the source IP address.
hash-source-port	(Optional) Defines the load-balancing hash of the source port.
l2-redirect	(Optional) Sets the packet forwarding by Layer 2 redirect. For more information, see the "WCCP Layer 2 Support" section on page 2-839.
mask-assign	(Optional) Uses the mask method for the Content Engine assignment.
password	(Optional) Sets the authentication password to be used for secure traffic among the Content Engines within a cluster and the router for a specified service. Be sure to enable all other Content Engines and routers within the cluster with the same password.
key	WCCP service password key. Passwords must not exceed eight characters.
weight	(Optional) Sets the weight percentage for load balancing. For more information, see the "Load Balancing" section on page 2-841.
percentage	Percentage value (0–100).

Defaults

wccp https-cache: disabled

dst-ip-mask: 0x00001741

src-ip-mask: 0x00000000

dst-port-mask: 0x0

src-port-mask: 0x0

Command Modes

global configuration command

Usage Guidelines

By default, the **wccp https-cache** command instructs the corresponding router to intercept port 443 TCP traffic and forward it to the Content Engine. However, the Content Engine will only accept the traffic if the HTTPS server is configured using the **https server** command.

The router administrator must use the **ip wccp 70** and **ip wccp 70 redirect out** commands on the redirect interface of the WCCP router.

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.

The Content Engine accepts redirected HTTPS traffic only if the HTTPS server was configured on the Content Engine (if you entered 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 and later releases, another interception mode (the accept-all mode) is available for the https-cache service (service 70). The accept-all mode was added to support the filtering of HTTPS traffic. 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).

WCCP service number 70 handles HTTPS traffic (the default port for such traffic is 443), and instructs the corresponding router to intercept port 443 TCP traffic and forward it to the Content Engine. The HTTPS service has a special feature that other WCCP services do not have. It uses an accept list of destination IP addresses to selectively redirect traffic to an application instead of redirecting all traffic to related applications. When the destination server's IP address in a client request matches any of the destination IP addresses in the accept list, WCCP redirects HTTPS traffic to the proper listening application. All other traffic is redirected to the corresponding router (normal bypass behavior). If static bypass is configured to allow traffic from specified sources to bypass the Content Engine, traffic is intercepted and is passed directly to the origin server. If an IP address is present on both the accept list and the bypass list, the Content Engine can return traffic to the WCCP-enabled router or switch and inform the router or switch to forward the packets as if the Content Engine were not present.

To intercept all HTTPS requests, use the **wccp https-cache accept-all** command regardless of whether the origin HTTPS servers are 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 (no SSL termination) the request to the origin HTTPS server. The Content Engine can negotiate an SSL connection with the client and serve HTTPS requests if the origin HTTPS server is configured on the Content Engine. Otherwise, HTTPS traffic is tunneled through the Content Engine (no SSL termination; the Content Engine passes traffic from client to server without modifying anything, except for filtering). When the SmartFilter software or Websense filtering software is enabled on the Content Engine, the Content Engine sends URLs corresponding to tunneled HTTPS traffic to the filtering server. Filtering servers might decide to block or pass the traffic. If traffic needs to be blocked, the Content Engine closes the connection with the client, and the request is not served.

To configure a Content Engine to support HTTPS transparent caching, you must configure the Content Engine and a router to support the WCCP Version 2 HTTPS caching service. 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.

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 "wccp service-number" section.

See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure HTTPS transparent caching on a Content Engine and a single router running WCCP Version 2.

Examples

The following example enables the Content Engine to accept redirected HTTPS requests from a WCCP-enabled router:

```
ContentEngine(config)# wccp version 2
ContentEngine(config)# wccp https-cache router-list-num 1
ContentEngine(config)# wccp router-list 1 172.16.202.1
```

The following example shows how to configure the WCCP-enabled router to support the https-cache service (service 70) by enabling WCCP Version 2 on the router (for example, Router A):

```
RouterA# configure terminal
RouterA(config)# ip wccp version 2
```

The following example shows how to configure Router A to run the https-cache service (service 70):

```
RouterA(config)# ip wccp 70
```

The following example shows how to configure Router A to intercept all outgoing HTTPS traffic:

```
Router(config)# ip wccp 70 redirect out
```

The following example shows how to configure the Content Engine to support the https-cache service by configuring the list of routers to support the https-cache service:

```
ContentEngine(config)# wccp router-list 1 10.1.202.1
```

In this example, router list number 1 is created and consists of only one router (Router A, which has an IP address of 10.2.202.1).

The following example shows how to 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 l2-redirect mask-assign
```

The **l2-redirect** option specifies Layer 2 redirection as the packet-forwarding method (instead of GRE). The **mask-assign** option specifies the mask assignment as the load-balancing method for this WCCP service.

The following example shows how to enable WCCP Version 2 on the Content Engine:

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

The following example shows how to enable the Content Engine to intercept all HTTPS traffic and tunnel the HTTPS traffic for which it does not have a private key or certificate and how to enable the accept-all mode on the Content Engine:

```
ContentEngine(config)# wccp https-cache accept-all
```

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).

The following example shows how to specify the server name of an origin HTTPS server from which the Content Engine caches the content:

```
ContentEngine(config)# https server abc1
ContentEngine(config-https)#
```

In the 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)#.

The following example shows how to enter the certificate, the private key, and the hostname of the HTTPS server (abc1), and then enable these settings (enter **enable** from the submode, as shown below) on the Content Engine from HTTPS configuration submode:

```
ContentEngine(config-https)# cert ?
ContentEngine(config-https)# cert mycert create
ContentEngine(config-https)# cert mycert import http://www.myca.com/myservercert
ContentEngine(config-https)# cert mykey create
ContentEngine(config-https)# cert mykey import http://www.myca.com/myprivatekey
ContentEngine(config-https)# host abc1
ContentEngine(config-https)# enable
```

These settings are the minimal settings for HTTPS that you need for enabling the content caching of the specified HTTPS server.

The **cert** and **key** command options configure a Content Engine to use a set of SSL certificates and keys that enables the Content Engine to act as the origin HTTPS server. The Content Engine is able to decode HTTPS traffic from a client and perform normal HTTP operations on it, such as caching and request processing. The Content Engine is able to initiate HTTPS connections to an origin server and fetch the content from origin servers upon a cache miss (or a cache validation). For more information, see the "https server" section on page 2-243.

The following example shows how to configure the Content Engine to use SSL Version 2 only either from HTTPS configuration submode or global configuration mode:

```
CONTENTENGINE(config-https)# protocol-version sslv2-only
```

or

```
CONTENTENGINE(config)# https server name protocol-version sslv2-only
```

Related Commands

https server
show https
show statistics https
show wccp content-engines
show wccp flows https-cache
show wccp masks https-cache
show wccp routers

show wccp slowstart https-cache
show wccp status
wccp version 2

wccp port-list

To associate ports with specific WCCP Version 2 dynamic services, use the **wccp port-list** global configuration command. To disassociate ports from WCCP Version 2 dynamic services, use the **no** form of this command.

wccp port-list *listnum portnum*

no wccp port-list *listnum portnum*

Syntax Description

listnum	Port list number (1–8).
portnum	Port number (1–65535). Up to eight ports per list number are allowed.

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

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.

Up to eight port numbers can be included in a single port list. The port list is referenced by the **wccp service-number** command that configures a specific WCCP Version 2 dynamic service (90–97) to operate on the listed ports.

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.

Examples

The following example shows that ports 10, 200, 3000, 110, 220, 330, 440, and 40000 are included in port list 3:

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

Related Commands

wccp service-number

wccp reverse-proxy

To enable WCCP Version 2 reverse proxy service, use the **wccp reverse-proxy** global configuration command. To disable this function, use the **no** form of this command.

wccp reverse-proxy {**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*]}

no wccp reverse-proxy

Syntax Description

mask	Sets the mask used for the Content Engine assignment. Configure at least one mask; the maximum is four masks.
dst-ip-mask	(Optional) Sets the mask used to match the packet destination IP address.
hex_num	IP address mask defined by a hexadecimal number (for example, 0xFC000000). The range is 0x00000000–FE000000.
dst-port-mask	(Optional) Sets the mask used to match the packet destination port number.
port_hex_num	Port mask defined by a hexadecimal number (for example, 0xFC00). The port range is 0–65535.
src-ip-mask	(Optional) Sets the mask used to match the packet source IP address.
src-port-mask	(Optional) Sets the mask used to match the packet source port number.
router-list-num	Sets the router list number.
num	Router list number (1–8).
assign-method-strict	(Optional) Forces WCCP to strictly use only the configured assignment method. For more information, see the "Assignment Method" section on page 2-840.
l2-redirect	(Optional) Sets the packet forwarding by Layer 2 redirect.
mask-assign	(Optional) Uses the mask method for the Content Engine assignment.
password	(Optional) Sets the authentication password.
key	WCCP service password key. Passwords must not exceed eight characters.
weight	(Optional) Sets the weight percentage for load balancing.
percentage	Percentage value (0–100).

Defaults

wccp reverse-proxy: disabled

dst-ip-mask: 0x00000000

src-ip-mask: 0x00001741

dst-port-mask: 0x0

src-port-mask: 0x0

Command Modes

global configuration

Usage Guidelines

This command applies only to WCCP Version 2.

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 Content Engine that is functioning as a transparent reverse proxy server.

You must configure the **wccp router-list** command before you use this command. The routers in the list must have WCCP reverse proxy service enabled (service 99).

By default, the router does load balancing across the various Content Engines in a cluster based on the destination IP address (for example, web server IP address). When WCCP reverse proxy is enabled, the router does load balancing in a cluster based on the source IP address (for example, the client's browser IP address).

To enable the use of a password for a secure reverse proxy cache within a cluster, use the **wccp reverse-proxy password** *key* command to be sure to enable all other Content Engines and routers within the cluster with the same password.

The **l2-redirect** option permits the Content Engine to receive transparently redirected traffic from a WCCP Version 2-enabled switch or router 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 the total load redirected to the Content Engine in a 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 a 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.

See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure the reverse-proxy service (service 99) on a Content Engine and a WCCP router.

Examples

The following example associates the router list 8 with the reverse-proxy service, sets the authentication password to be used for secure traffic among the Content Engines within a cluster and the router for the reverse-proxy service, and sets the weight percentage for load balancing:

```
ContentEngine(config)# wccp reverse-proxy router-list-num 8 password mysecret weight 100
```

The following example disables the reverse-proxy service:

```
ContentEngine(config)# no wccp reverse-proxy
```

The following example shows how to configure the Content Engine to support the reverse-proxy service by enabling WCCP Version 2 on the 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.

The following example shows how to create a router list that specifies the routers that will support the reverse-proxy service:

```
ContentEngine(config)# wccp router-list 1 10.0.1.1
```

The IP address or multicast address specifies which router 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).

The following example shows how to inform the WCCP-enabled router in the specified router list that this Content Engine is accepting redirected reverse proxy cache requests on port 80:

```
ContentEngine(config)# wccp reverse-proxy router-list-num 1 l2-redirect mask-assign
```

The **l2-redirect** option specifies Layer 2 redirection as the packet-forwarding method (instead of GRE). The **mask-assign** option specifies the mask assignment as the load-balancing method for this WCCP service.

The following example shows how to write the running configurations to nonvolatile memory:

```
ContentEngine# write memory
```

The following example shows how to configure a WCCP-enabled router to support the reverse-proxy service by turning on WCCP Version 2 on the router:

```
Router# configure terminal
Router(config)# ip wccp version 2
```

The following example shows how to turn on the reverse-proxy service (service 99) on the router:

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

The following example shows how to specify the interface on which the reverse-proxy service will run:

```
Router(config)# interface ethernet 0
```

In the example, the Ethernet 0 interface on the router is configured to run the reverse-proxy service.

The following example shows how to configure the router to use the outbound interface for the reverse-proxy service:

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

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 the (transparent reverse proxy server).

Related Commands

show wccp content-engines
show wccp flows reverse-proxy
show wccp masks reverse-proxy
show wccp routers
show wccp services
show wccp slowstart reverse-proxy
show wccp status
wccp router-list
wccp version 2

wccp router-list

To configure a router list for WCCP Version 2, use the **wccp router-list** global configuration command. To disable this function, use the **no** form of this command.

wccp router-list *number ip-address*

no wccp router-list *number ip-address*

Syntax Description

number	Router list number (1–8).
ip-address	IP address of router to add to the list.

Defaults

Disabled

Command Modes

global configuration

Usage Guidelines

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.

Use the **wccp router-list** command to configure various router lists for use with WCCP Version 2 services. 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. When you create a router list, up to six IP addresses can be added per line. Multiple lines can be used to represent a router list with a maximum of 32 routers. For example, you can specify one router list for the WCCP Version 2 web cache service and another list for reverse proxy at the same time without having to reconfigure groups of routers or Content Engines. You can add up to 8 router lists and up to 32 IP addresses per list.

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.

Note You must enter the **ip wccp** global configuration command 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. Using a list of routers on the Content Engine precludes the need to use IP multicast but requires more configuration on each Content Engine. Using an IP multicast address reduces the configuration on the Content Engine and 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.

Additional configuration is 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:

- You must configure the IP multicast address for use by the service group.
- You need to configure the interface or interfaces so that the WCCP Version 2-enabled router can receive the IP multicast address by entering the **ip wccp** {**web-cache** | *service-number*} **group-listen** command.

For the network configurations in which another router must be traversed to get to the target router, you must configure the router being traversed to perform IP multicast routing as follows:

- Enable IP multicast routing by entering the **ip multicast-routing** command.
- Configure the router interfaces that connect to the Content Engines to receive multicasting by entering the **ip pim** command.

Examples

The following example shows how router list number 7 is created and contains a single router (the WCCP Version 2-enabled router with IP address 172.31.68.98):

```
ContentEngine(config)# wccp router-list 7 172.31.68.98
```

The following example deletes the router list number 7 created in the previous example:

```
ContentEngine(config)# no wccp router-list 7 172.31.68.98
```

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
```

Related Commands

wccp reverse-proxy
wccp version 2
wccp web-cache