clock (EXEC)

To set or clear clock functions or update the calendar, use the **clock** EXEC command.

clock {**read-calendar** | **set** *time day month year* | **update-calendar**}

Syntax Description

read-calendar	Reads the calendar and updates the system clock.
set	Sets the time and date.
time	Current time in hh:mm:ss format (hh: 00–23; mm: 00–59; ss: 00–59).
day	Day of the month (1–31).
month	Month of the year (January, February, March, April, May, June, July, August, September, October, November, December).
year	Year (1993–2035).
update-calendar	Updates the calendar with the system clock.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

If you have an outside source on your network that provides time services (such as a Network Time Protocol [NTP] server), you do not need to set the system clock manually. When setting the clock, enter the local time. The Content Engine calculates the Coordinated Universal Time (UTC) based on the time zone set by the **clock timezone** global configuration command.

Note We strongly recommend that you configure the Content Engine for the Network Time Protocol (NTP) by using the **ntp** global configuration command. See the "ntp" section on page 2-356 for more details.

Two clocks exist in the system: the software clock and the hardware clock. The software uses the software clock. The hardware clock is used only at bootup to initialize the software clock. The calendar clock is the same as the hardware clock that runs continuously on the system, even if the system is powered off or rebooted. This clock is separate from the software clock settings, which are erased when the system is powered cycled or rebooted.

The **set** keyword sets the software clock. If the system is synchronized by a valid outside timing mechanism, such as a Network Time Protocol (NTP) clock source, you do not need to set the system clock. Use this command if no other time sources are available. The time specified in this command is relative to the configured time zone.

To perform a one-time update of the hardware clock (calendar) from the software clock or to copy the software clock settings to the hardware clock (calendar), use the **clock update-calendar** EXEC command.

The Content Engine Network Module (CE-NM) causes the real-time clock on the board to be reset when it is powered off for more than 10 minutes. As a result, the clock on the CE-NM might be reset to 1980 if it is powered off for a long period. Several applications that depend on the correct time being configured on the Network Module might not work in such a scenario.

Note We strongly recommend that you configure the CE-NM for the NTP using the **ntp server** {*ip-address* | *hostname*} global configuration command, either after an upgrade from the ACNS 4.2.x software to the ACNS 5.x software, or on obtaining a factory-fresh CE-NM, to maintain the correct time on the Network Module. This configuration ensures that the system clock on the CE-NM is always synchronized with the NTP time server's clock.

Examples

The following example sets the software clock on the Content Engine:

```
ContentEngine# clock set 13:32:00 01 February 2000
```

Related Commands

clock timezone
ntp
show clock detail

clock (global configuration)

To set the summer daylight saving time and time zone for display purposes, use the **clock** global configuration command. To disable this function, use the **no** form of this command.

clock {**summertime** *timezone* {**date** *startday startmonth startyear starthour endday endmonth endyear offset* | **recurring** {**1-4** *startweekday startmonth starthour endweekday endmonth endhour offset* | **first** *startweekday startmonth starthour endweekday endmonth endhour offset* | **last** *startweekday startmonth starthour endweekday endmonth endhour offset*}} | **timezone** {*timezone hoursoffset minutesoffset*}}

no clock {**summertime** *timezone* {**date** *startday startmonth startyear starthour endday endmonth endyear offset* | **recurring** {**1-4** *startweekday startmonth starthour endweekday endmonth endhour offset* | **first** *startweekday startmonth starthour endweekday endmonth endhour offset* | **last** *startweekday startmonth starthour endweekday endmonth endhour offset*}} | **timezone** {*timezone hoursoffset minutesoffset*}}

Syntax Description

summertime	Configures the summer or daylight saving time.
timezone	Name of the summer time zone.
date	Configures the absolute summer time.
startday	Date (1–31) to start.
startmonth	Month (January through December) to start.
startyear	Year (1993–2032) to start.
starthour	Hour (0–23) to start in (hh:mm) format.
endday	Date (1–31) to end.
endmonth	Month (January through December) to end.
endyear	Year (1993–2032) to end.
endhour	Hour (0–23) to end in (hh:mm) format.
offset	Minutes offset (see Table A-1 on page A-1) from Coordinated Universal Time (UTC) (0–59).
recurring	Configures the recurring summer time.
1-4	Configures the starting week number 1–4.
first	Configures the summer time to recur beginning the first week of the month.
last	Configures the summer time to recur beginning the last week of the month.
startweekday	Day of the week (Monday–Friday) to start.
startmonth	Month (January–December) to start.
starthour	Hour (0–23) to start in (hh:mm) format.
endweekday	Weekday (Monday–Friday) to end.
endmonth	Month (January–December) to end.
endhour	Hour (0–23) to end in hour:minute (hh:mm) format.
offset	Minutes offset (see Table A-1 on page A-1) from UTC (0–59).
timezone	Configures the standard time zone.
timezone	Name of the time zone.

hoursoffset	Hours offset (see Table A-1 on page A-1) from UTC (–23 to +23).
minutesoffset	Minutes offset (see Table A-1 on page A-1) from UTC (0–59).

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

To set and display the local and UTC current time of day without an NTP server, use the **clock timezone** command with the **clock set** command. The **clock timezone** parameter specifies the difference between UTC and local time, which is set with the **clock set** EXEC command. The UTC and local time are displayed with the **show clock detail** EXEC command.

Use the **clock** *timezone offset* command to specify a time zone, where *timezone* is the desired time zone entry from Table A-1 on page A-1 and *0 0* is the offset (ahead or behind) Coordinated Universal Time (UTC) in hours and minutes. UTC was formerly known as Greenwich mean time (GMT).

```
CE(config)# clock timezone timezone 0 0
```

Note The time zone entry is case sensitive and must be specified in the exact notation listed in the time zone table as shown in Appendix B, "Standard Time Zones." When you use a time zone entry from Table A-1 on page A-1, the system is automaticaly adjusted for daylight saving time.

The offset (ahead or behind) UTC in hours, as displayed in Table A-1 on page A-1, is in effect during winter time. During summer time or daylight saving time, the offset may be different from the values in the table and are calculated and displayed accordingly by the system clock.

Note An accurate clock and timezone setting is required for the correct operation of the HTTP proxy chaches.

Examples

The following example specifies the local time zone as Pacific Standard Time with an offset of 8 hours behind UTC:

```
ContentEngine(config)# clock timezone PST -8
Custom Timezone: PST will be used.
```

The following example configures a standard time zone on the Content Engine:

```
CONTENTENGINE(config)# clock timezone US/Pacific 0 0
Resetting offset from 0 hour(s) 0 minute(s) to -8 hour(s) 0 minute(s)
Standard Timezone: US/Pacific will be used.
CONTENTENGINE(config)#
```

The following example negates the time zone setting on the Content Engine:

```
ContentEngine(config)# no clock timezone
```

The following example configures daylight saving time:

```
ContentEngine(config)# clock summertime PDT date 10 October 2001 23:59 29 April 2002 23:59
60
```

Related Commands

clock
show clock detail

cms (EXEC)

To configure the Centralized Management System (CMS) embedded database parameters, use the **cms** EXEC command.

cms {**config-sync** | **database** {**backup** | **create** | **delete** | **downgrade** [**script** *filename*] | **lcm** {**enable** | **disable**} | **maintenance** {**full** | **regular**} | **restore** *filename* | **validate**} | **deregister** [**force**] | **recover** {**identity** *word*}}

Syntax Description

config-sync	Sets the node to synchronize configuration with the Content Distribution Manager.
database	Creates, backs up, deletes, restores, or validates the CMS-embedded database management tables or files.
backup	Backs up the database management tables.
create	Creates the embedded database management tables.
delete	Deletes the embedded database files.
downgrade	Downgrades the CMS database.
script	(Optional) Downgrades the CMS database by applying a downgrade script.
filename	Downgraded script filename.
lcm	Configures local/central management on an ACNS network device that is registered with the Content Distribution Manager.
enable	Enables synchronization of the ACNS network configuration of the device with the local CLI configuration.
disable	Disables synchronization of the ACNS network configuration of the device with the local CLI configuration.
maintenance	Cleans and reindexes the embedded database tables.
full	Specifies a full maintenance routine for the embedded database tables.
regular	Specifies a regular maintenance routine for the embedded database tables.
restore	Restores the database management tables using the backup local filename.
filename	Database local backup filename.
validate	Validates the database files.
deregister	Removes the registration of the CMS proto device.
force	(Optional) Forces the removal of the node registration.
recover	Recovers the identity of an ACNS network device.
identity	Specifies the identity of the recovered device.
word	Identity of the recovered device.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

The ACNS network is a collection of Content Router, Content Engine, and Content Distribution Manager nodes. One primary Content Distribution Manager retains the ACNS network settings and provides other ACNS network nodes with updates. Communication between nodes occurs over secure channels using the Secure Shell Layer (SSL) protocol, where each node on the ACNS network uses a Rivest, Shamir, Adelman (RSA) certificate-key pair to communicate with other nodes.

Use the **cms config-sync** command to enable registered Content Routers, Content Engines, and standby Content Distribution Manager to contact the primary Content Distribution Manager immediately for a getUpdate (get configuration poll) request before the default polling interval of 5 minutes. For example, when a node is registered with the primary Content Distribution Manager and activated, it appears as Pending in the Content Distribution Manager GUI until it sends a getUpdate request. The **cms config-sync** command causes the registered node to send a getUpdate request at once, and the status of the node changes as Online.

Use the **cms database create** command to initialize the CMS database. Before a node can join an ACNS network, it must first be registered and then activated. The **cms enable** global configuration command automatically registers the node in the database management tables and enables the CMS. The node sends its attribute information to the Content Distribution Manager over the SSL protocol and then stores the new node information. The Content Distribution Manager accepts these node registration requests without admission control and replies with registration confirmation and other pertinent security information required for getting updates. Activate the node using the Content Distribution Manager GUI.

Once the node is activated, it automatically receives configuration updates and the necessary security RSA certificate-key pair from the Content Distribution Manager. This security key allows the node to communicate with any other node in the ACNS network. The **cms deregister** command removes the node from the ACNS network by deleting registration information and database tables.

To back up the existing management database for the Content Distribution Manager, use the **cms database backup** command. For database backups, specify the following items:

- Location, password, and user ID
- Dump format in PostgreSQL plain text syntax

The naming convention for backup files includes the time stamp.

Note For information on the procedure to back up and restore the CMS database on the Content Distribution Manager, see the *Cisco ACNS Software Upgrade and Maintenance Guide*.

When you use the **cms recover identity** *word* command when recovering lost registration information, or replacing a failed node with a new node that has the same registration information, you must specify the device recovery key that you configured in the Modifying Config Property, System.device.recovery.key window of the Content Distribution Manager GUI.

Use the **lcm** command to configure local/central management (LCM) on an ACNS network device. The LCM feature allows settings configured using the device CLI or GUI to be stored as part of the ACNS network-wide configuration data (enable or disable).

When you enter the **cms lcm enable** command, the CMS process running on Content Engines, Content Routers, and the standby Content Distribution Manager detects the configuration changes that you made on these devices using CLIs and sends the changes to the primary Content Distribution Manager.

When you enter the **cms lcm disable** command, the CMS process running on Content Engines, Content Routers, and the standby Content Distribution Manager does not send the CLI changes to the primary Content Distribution Manager. Settings configured using the device CLIs will not be sent to the primary Content Distribution Manager.

If LCM is disabled, the settings configured through the Content Distribution Manager GUI will overwrite the settings configured from the Content Engine or Content Router; however, this rule applies only to those local device settings that have been overwritten by the Content Distribution Manager when you have configured the local device settings. If you (as the local CLI user) change the local device settings after the particular configuration has been overwritten by the Content Distribution Manager, the local device configuration will be applicable until the Content Distribution Manager requests a full device statistics update from the Content Engine or Content Router (clicking the **Force full database update** button from the Device Home window of the Content Distribution Manager GUI triggers a full update). When the Content Distribution Manager requests a full update from the device, the Content Distribution Manager settings will overwrite the local device settings.

Examples

The following example backs up the database management tables:

```
ContentDistributionManager# cms database backup
creating backup file with label `backup'
backup file local1/acns-db-9-22-2002-17-36.dump is ready. use `copy' commands to move the
backup file to a remote host.
```

The following example validates the database management tables:

```
ContentDistributionManager# cms database validate
Management tables are valid
```

In the following example, the CMS deregistration process has problems deregistering the Content Engine, but it proceeds to deregister it from the CMS database when the **force** option is used:

```
ContentEngine# cms deregister force
Deregistration requires management service to be stopped.
You will have to manually start it. Stopping management service on this node...
This operation needs to restart http proxy and streaming proxies/servers (if running) for
memory reconfiguration. Proceed? [no]yes
management services stopped
Thu Jun 26 13:17:34 UTC 2003 [I] main: creating 24 messages
Thu Jun 26 13:17:34 UTC 2003 [I] main: creating 12 dispatchers
Thu Jun 26 13:17:34 UTC 2003 [I] main: sending eDeRegistration message to CDM
10.107.192.168
...
ContentEngine#
```

The following example shows the use of the **cms recover identity** command when the recovery request matches the Content Engine record, and the Content Distribution Manager updates the existing record and sends a registration response to the requesting Content Engine:

```
ContentEngine# cms recover identity default
Registering this node as Content Engine...
Sending identity recovery request with key default
Thu Jun 26 12:54:42 UTC 2003 [I] main: creating 24 messages
Thu Jun 26 12:54:42 UTC 2003 [I] main: creating 12 dispatchers
Thu Jun 26 12:54:42 UTC 2003 [I] main: Sending registration message to CDM  10.107.192.168
Thu Jun 26 12:54:44 UTC 2003 [W] main: Unable to load device info file in TestServer
Thu Jun 26 12:54:44 UTC 2003 [I] main: Connecting storeSetup for CE.
Thu Jun 26 12:54:44 UTC 2003 [I] main: Instantiating AStore
'com.cisco.unicorn.schema.PSqlStore'...
Thu Jun 26 12:54:45 UTC 2003 [I] main: Successfully connected to database
Thu Jun 26 12:54:45 UTC 2003 [I] main: Registering object factories for persistent
store...
Thu Jun 26 12:54:51 UTC 2003 [I] main: Dropped Sequence IDSET.
Thu Jun 26 12:54:51 UTC 2003 [I] main: Successfully removed old management tables
Thu Jun 26 12:54:51 UTC 2003 [I] main: Registering object factories for persistent
store...
```

```
Thu Jun 26 12:54:51 UTC 2003 [I] main: Creating PSql Table BYPASS_INFO
.
.
.
Thu Jun 26 12:54:54 UTC 2003 [I] main: Created Table FILE_CDM.
Thu Jun 26 12:54:55 UTC 2003 [I] main: Created SYS_MESS_TIME_IDX index.
Thu Jun 26 12:54:55 UTC 2003 [I] main: Created SYS_MESS_NODE_IDX index.
Thu Jun 26 12:54:55 UTC 2003 [I] main: No Consistency check for store.
Thu Jun 26 12:54:55 UTC 2003 [I] main: Successfully created management tables
Thu Jun 26 12:54:55 UTC 2003 [I] main: Registering object factories for persistent
store...
Thu Jun 26 12:54:55 UTC 2003 [I] main: AStore Loading store data...
Thu Jun 26 12:54:56 UTC 2003 [I] main: ExtExpiresRecord Loaded 0 Expires records.
Thu Jun 26 12:54:56 UTC 2003 [I] main: Skipping Construction RdToClusterMappings on
non-CDM node.
Thu Jun 26 12:54:56 UTC 2003 [I] main: AStore Done Loading. 327
Thu Jun 26 12:54:56 UTC 2003 [I] main: Created SYS_MESS_TIME_IDX index.
Thu Jun 26 12:54:56 UTC 2003 [I] main: Created SYS_MESS_NODE_IDX index.
Thu Jun 26 12:54:56 UTC 2003 [I] main: No Consistency check for store.
Thu Jun 26 12:54:56 UTC 2003 [I] main: Successfully initialized management tables
Node successfully registered with id 103
Registration complete.
ContentEngine#
```

The following example shows the use of the **cms recover identity** command when the hostname of the Content Engine does not match the hostname configured in the Content Distribution Manager graphical user interface:

```
ContentEngine# cms recover identity default
Registering this node as Content Engine...
Sending identity recovery request with key default
Thu Jun 26 13:16:09 UTC 2003 [I] main: creating 24 messages
Thu Jun 26 13:16:09 UTC 2003 [I] main: creating 12 dispatchers
Thu Jun 26 13:16:09 UTC 2003 [I] main: Sending registration message to CDM  10.107.192.168
There're no CE devices in CDN
register: Registration failed.
ContentEngine#
```

Related Commands

cms enable
show cms

cms (global configuration)

To schedule maintenance and enable the Centralized Management System (CMS) on a given node, use the **cms** global configuration command. To negate these actions, use the **no** form of this command.

cms {**database maintenance** {**full** {**enable** | **schedule** *weekday* **at** *time*} | **regular** {**enable** | **schedule** *weekday* **at** *time*}} | **enable** | **rpc timeout** {**connection** *5-1800* | **incoming-wait** *10-600* | **transfer** *10-7200*}}

no cms {**database maintenance** {**full** {**enable** | **schedule** *weekday* **at** *time*} | **regular** {**enable** | **schedule** *weekday* **at** *time*}} | **enable** | **rpc timeout** {**connection** *5-1800* | **incoming-wait** *10-600* | **transfer** *10-7200*}}

Syntax Description

database maintenance	Configures the embedded database clean or reindex maintenance routine.
full	Configures the full maintenance routine and cleans the embedded database tables.
enable	Enables the full maintenance routine to be performed on the embedded database tables.
schedule	Sets the schedule for performing the maintenance routine.
weekday	Day of the week to start the maintenance routine. every-day Every day Mon every Monday Tue every Tuesday Wed every Wednesday Thu every Thursday Fri every Friday Sat every Saturday Sun every Sunday
at	Sets the maintenance schedule time of day to start the maintenance routine.
time	Time of day to start the maintenance routine (0–23:0–59) (hh:mm). at Maintenance time of day Mon every Monday Tue every Tuesday Wed every Wednesday Thu every Thursday Fri every Friday Sat every Saturday Sun every Sunday
regular	Configures the regular maintenance routine and reindexes the embedded database tables.
enable	Enables the node CMS process.
rpc timeout	Configures the timeout values for remote procedure call connections.
connection	Specifies the maximum time to wait when making a connection.
5-1800	Timeout period in seconds. The default for the Content Distribution Manager is 30 seconds; the default for the Content Engine and the Content Router is 180 seconds.
incoming-wait	Specifies the maximum time to wait for a client response.

10-600	Timeout period in seconds. The default is 30 seconds.
transfer	Specifies the maximum time to allow a connection to remain open.
10-7200	Timeout period in seconds. The default is 300 seconds.

Defaults

database maintenance regular: enabled

database maintenance full: enabled

connection: 30 seconds for Content Distribution Manager; 180 seconds for the Content Engine and the Content Router

incoming wait: 30 seconds

transfer: 300 seconds

Command Modes

global configuration

Usage Guidelines

Use the **cms database maintenance** command to schedule routine full maintenance cleaning (vacuuming) or a regular maintenance reindexing of the embedded database. The full maintenance routine runs only when the disk is more than 90 percent full and only runs once a week. Cleaning the tables returns reusable space to the database system.

The **cms enable** command automatically registers the node in the database management tables and enables the CMS process. The **no cms enable** command only stops the management services on the device and does not disable a primary sender. You can use the **cms deregister** command to remove a primary or backup sender Content Engine from the ACNS network and to disable communication between the two multicast senders.

Examples

The following example schedules a regular (reindexing) maintenance routine to start every Friday at 11:00 p.m.:

```
ContentEngine(config)# cms database maintenance regular schedule Fri at 23:00
```

The following example shows how to enable the CMS process on a Content Engine:

```
ContentEngine(config)# cms enable
This operation needs to restart http proxy and streaming proxies/servers (if running) for
memory reconfiguration. Proceed? [no]yes
Registering this node as Content Engine...
Thu Jun 26 13:18:24 UTC 2003 [I] main: creating 24 messages
Thu Jun 26 13:18:25 UTC 2003 [I] main: creating 12 dispatchers
Thu Jun 26 13:18:25 UTC 2003 [I] main: Sending registration message to CDM  10.107.192.168
Thu Jun 26 13:18:27 UTC 2003 [I] main: Connecting storeSetup for CE.
Thu Jun 26 13:18:27 UTC 2003 [I] main: Instantiating AStore
'com.cisco.unicorn.schema.PSqlStore'...
Thu Jun 26 13:18:28 UTC 2003 [I] main: Successfully connected to database
Thu Jun 26 13:18:28 UTC 2003 [I] main: Registering object factories for persistent
store...
Thu Jun 26 13:18:35 UTC 2003 [I] main: Dropped Sequence IDSET.
Thu Jun 26 13:18:35 UTC 2003 [I] main: Dropped Sequence GENSET.
Thu Jun 26 13:18:35 UTC 2003 [I] main: Dropped Table USER_TO_DOMAIN.
.
.
.
```

```
Thu Jun 26 13:18:39 UTC 2003 [I] main: Created Table FILE_CDM.
Thu Jun 26 13:18:40 UTC 2003 [I] main: Created SYS_MESS_TIME_IDX index.
Thu Jun 26 13:18:40 UTC 2003 [I] main: Created SYS_MESS_NODE_IDX index.
Thu Jun 26 13:18:40 UTC 2003 [I] main: No Consistency check for store.
Thu Jun 26 13:18:40 UTC 2003 [I] main: Successfully created management tables
Thu Jun 26 13:18:40 UTC 2003 [I] main: Registering object factories for persistent
store...
Thu Jun 26 13:18:40 UTC 2003 [I] main: AStore Loading store data...
Thu Jun 26 13:18:41 UTC 2003 [I] main: ExtExpiresRecord Loaded 0 Expires records.
Thu Jun 26 13:18:41 UTC 2003 [I] main: Skipping Construction RdToClusterMappings on
non-CDM node.
Thu Jun 26 13:18:41 UTC 2003 [I] main: AStore Done Loading. 336
Thu Jun 26 13:18:41 UTC 2003 [I] main: Created SYS_MESS_TIME_IDX index.
Thu Jun 26 13:18:41 UTC 2003 [I] main: Created SYS_MESS_NODE_IDX index.
Thu Jun 26 13:18:41 UTC 2003 [I] main: No Consistency check for store.
Thu Jun 26 13:18:41 UTC 2003 [I] main: Successfully initialized management tables
Node successfully registered with id 28940
Registration complete.
Warning: The device will now be managed by the CDM. Any configuration changes
made via CLI on this device will be overwritten if they conflict with settings on the CDM.
Please preserve running configuration using 'copy running-config startup-config'.
Otherwise management service will not be started on reload and node will be shown
'offline' in CDM UI.
management services enabled
ContentEngine(config)#
```

Related Commands

cms database
cms deregister
show cms

configure

To enter global configuration mode, use the **configure** EXEC command. You must be in global configuration mode to enter global configuration commands.

configure

To exit global configuration mode, use the **end** or **exit** commands. In addition, you can press **Ctrl-Z** to exit from global configuration mode.

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to enter global configuration mode.

Examples

The following example shows how to enable global configuration mode:

```
ContentEngine# configure
ContentEngine(config)#
```

Related Commands

end
exit
show running-config
show startup-config

contentrouting

To configure dynamic and load-based content routing on the Content Router and to configure service monitor options on the Content Engine, use the **contentrouting** global configuration command. The service monitor options provide inputs to the Content Router for load-based content routing. Use the **no** form of this command to disable this feature.

contentrouting {**dynamic** | **leastloaded**}

contentrouting servicemonitor {**numberofsamples** {**all** *1-120* | **cpu** *1-120* | **disk** *1-120* | **wmt** *1-120*} | **sampleperiod** {**all** *1-60* | **cpu** *1-60* | **disk** *1-60* | **wmt** *1-60*} | **threshold** {**cpu** *1-100* | **wmt** *1-100* | **type** {**all** | **cpu** | **disk** | **wmt**}}

Syntax Description

dynamic	Enables dynamic content routing on the Content Router.
leastloaded	Enables load-based content routing on the Content Router.
servicemonitor	Specifies the Content Engine load-monitoring module. This module now enables the Content Engine to Content Router keepalive mechanism to carry load information.
numberofsamples	Specifies the number of latest sampled values to be used when calculating the average.
all	Specifies that the **numberofsamples** or **sampleperiod** pertains to the CPU, disk, and WMT data.
cpu	Specifies that the **numberofsamples** or **sampleperiod** pertains to the CPU data.
disk	Specifies that the **numberofsamples** or **sampleperiod** pertains to the disk data.
wmt	Specifies that the **numberofsamples** or **sampleperiod** pertains to the data about the active streams.
1-120	Number of latest monitored sample values to be used in calculating the average. The default value is 60.
sampleperiod	Specifies the time interval, in seconds, between two consecutive samples.
1-60	Sample period in seconds. The default is 5 seconds.
threshold	Allows the configuration of a threshold, in percentages, for the monitored data type. The Content Router does not route requests to the Content Engine that has exceeded this threshold.
1-100	Threshold value, in percentages. The default is 90 percent for the WMT-stream and 80 percent for the CPU.
type	Specifies the type of data that is to be monitored.

Defaults

dynamic: disabled

leastloaded: disabled

servicemonitor: disabled.

Command Modes

global configuration

Usage Guidelines

Dynamic Content Routing

In previous releases of the ACNS software, Content Routers used a static coverage zone file to describe the preferred routing path between Content Engines and client-end systems.

A coverage zone is a mapping of client end-system IP addresses to Content Engines. The Content Router uses the Content Engine IP addresses to create a static redirection table that maps end-system IP addresses to Content Engines and provides information on the proximity of end systems to Content Engines. When content is requested by a client, the Content Router checks the client IP address to find the coverage zone that contains that IP address. The Content Router then selects the Content Engine that is serving this coverage zone.

In some ACNS network environments, Content Engine IP addresses keep changing, and coverage zones are dynamic instead of static. In such cases, the Content Router cannot create a static routing table, and it cannot successfully route the content.

When the following conditions are present, the content cannot be routed successfully by using static coverage zone tables in the Content Router:

- Multiple Content Engines are deployed in multiple locations.
- Each location contains a NAT firewall.
- One Content Router serves all locations.
- One root Content Engine serves all locations.
- Each location is configured with two uplink lines to the Internet for redundancy.
- Uplink lines for different locations can share an external public IP address pool so that the same IP address can be used by NAT firewalls in different locations at different times.

With multiple uplinks to the Internet, requests for content from clients and Content Engines that are in the same location can go out to the Content Router with different external IP addresses. The Content Router that is using static coverage zone files cannot share the same IP address pool among different locations.

In the ACNS 5.3.3 software and later releases, the Content Router can detect the changes in Content Engine coverage zones and can dynamically adjust its routing tables. Use the **contentrouting dynamic** command to enable dynamic content routing on the Content Router.

Load Based Content Routing

In the ACNS 5.4 software and later releases, load based content routing has been enabled. The Content Router redirects client requests to the Content Engine that reports the lowest average CPU load, given that each Content Engine in the routing table has the same metric value (or weight).

When you have multiple Content Engines that are defined with unequal weighting, you can configure threshold limits for CPU load, disk usage, and WMT stream count, so that the Content Router redirects the client requests to the next preferred Content Engine that has not exceeded its threshold.

When a configured threshold is exceeded, messages are sent to the Content Router. Content Router algorithms compare the Content Engine assigned weight and current load to the configured threshold values in making routing decisions. Use the **contentrouting leastloaded** command to enable load-based content routing on the Content Router.

Load Monitoring Module

Use the **contentrouting sevicemonitor** command to enable load monitoring on the Content Engine. This module allows you to monitor the CPU load, and disk I/O statistics, and WMT stream count, which involves calculating a moving average for the CPU load percentage and disk I/O average queue size and getting the current active WMT stream count. The current WMT stream count is used to compare the

number of current actual streams to the license limits or configured limits for the Content Engine. It may be possible for the CPU to be low, but the Content Engine may not be allowed to serve additional streams due to license limits. The CPU-busy moving average is compared to a threshold for each routing keepalive update period.

The average queue size of a disk is monitored for disk I/O statistics. The monitored value is normalized by scaling to a factor of five to convert the value to a percentage. This normalized value is used for calculating the moving average.

Examples

The following example shows a sample configuration of the load-monitoring module:

```
ContentEngine(Config)# contentrouting servicemonitor type cpu
ContentEngine(Config)# contentrouting servicemonitor sampleperiod 5
ContentEngine(Config)# contentrouting servicemonitor threshold cpu 80
```

This configuration results in the CPU load being monitored. The moving average uses the latest 60 sample values (default); the samples are obtained every 5 seconds. If the CPU's moving average exceeds 80 percent, the CPU ThresholdExceed state is sent to the Content Router for the polling period under consideration.

Related Commands

show content-routing

content-routing-api

To configure the content routing API, use the **content-routing-api** global configuration command. Use the **no** form of this command to disable this feature. This command is available only on the Content Router.

content-routing-api enable

no content-routing-api enable

Syntax Description

enable	Enables the content routing API.

Defaults

Content routing API is disabled by default.

Command Modes

global configuration

Usage Guidelines

The Content Router uses HTTP or RTSP redirection to route requests to the Content Engine, which the Content Router determines is best suited to deliver the desired content to the client. This determination is based on the requested FQDN, the client's IP address, the assignment of a Content Engine to a website, and the keepalive probes sent by a Content Engine. The content routing API enables clients to query the Content Router's routing decision based on a combination of a client IP address and a website FQDN.

HTTP clients can make API calls to the Content Router on port 8888. You must configure the port number on which the Content Router accepts incoming requests from HTTP clients in addition to configuring port 80 using the **http proxy incoming** *port_num* global configuration command.

Note For more information about the content routing API, see the *Cisco ACNS Software API Guide*.

Examples

The following example enables the Content Routing API:

```
ContentRouter(config)# content-routing-api enable
```

Related Commands

http proxy incoming

copy

To copy the configuration or image data from a source to a destination, use the **copy** EXEC command.

copy cdnfs disk *url sysfs-filename*

copy cdrom install *filedir filename*

copy compactflash install *filename*

copy disk {**ftp** {*hostname* | *ip-address*} *remotefiledir remotefilename localfilename* | **startup-config** *filename*}

copy ftp {**disk** {*hostname* | *ip-address*} *remotefiledir remotefilename localfilename* | **install** {*hostname* | *ip-address*} *remotefiledir remotefilename*}

copy http install {{*hostname* | *ip-address*} *remotefiledir remotefilename*} [**port** *port-num* [**proxy** {*hostname* | *ip-address*} | **username** *username password* [**proxy** {*hostname* | *ip-address*} *proxy_portnum*]] | **proxy** {*hostname* | *ip-address*} *proxy_portnum* | **username** *username password* [**proxy** {*hostname* | *ip-address*} *proxy_portnum*]]

copy running-config {**disk** *filename* | **startup-config** | **tftp** {*hostname* | *ip-address*} *remotefilename*}

copy startup-config {**disk** *filename* | **running-config** | **tftp** {*hostname* | *ip-address*} *remotefilename*}

copy system-status disk *filename*

copy tech-support {**disk** *filename* | **tftp** {*hostname* | *ip-address*} *remotefilename*}

copy tftp {**disk** {*hostname* | *ip-address*} *remotefilename localfilename* | **running-config** {*hostname* | *ip-address*} *remotefilename* | **startup-config** {*hostname* | *ip-address*} *remotefilename*}

Syntax Description

cdnfs	Copies a file from the cdnfs to the sysfs.
disk	Copies a file to the disk.
url	URL of the cdnfs file to be copied to the sysfs.
sysfs-filename	Filename to be copied in the sysfs.
cdrom	Copies a file from the CD-ROM.
install	Installs the software release file.
filedir	Directory location of the software release file.
filename	Filename of the software release file.
compactflash	Copies a file from the CompactFlash card.
install	Installs a software release file.
filename	Image filename.
disk	Copies a local disk file.
ftp	Copies to a file on an FTP server.
hostname	Hostname of the FTP server.

ip-address	IP address of the FTP server.
remotefiledir	Directory on the FTP server to which the local file is copied.
remotefilename	Name of the local file once it has been copied to the FTP server.
localfilename	Name of the local file to be copied.
startup-config	Copies the configuration file from the disk to startup configuration (NVRAM).
filename	Name of the existing configuration file.
ftp	Copies a file from an FTP server.
disk	Copies a file to a local disk.
hostname	Hostname of the FTP server.
ip-address	IP address of the FTP server.
remotefiledir	Directory on the FTP server where the file to be copied is located.
remotefilename	Name of the file to be copied to the local disk.
localfilename	Name of the copied file as it appears on the local disk.
install	Copies the file from an FTP server and installs the software release file to the local device.
hostname	Name of the FTP server.
ip-address	IP address of the FTP server.
remotefiledir	Remote file directory.
remotefilename	Remote filename.
http install	Copies the file from an HTTP server and installs the software release file on a local device.
hostname	Name of the HTTP server.
ip-address	IP address of the HTTP server.
remotefiledir	Remote file directory.
remotefilename	Remote filename.
port	(Optional) Specifies the port to connect to the HTTP server (default is 80).
port-num	HTTP server port number (1–65535).
proxy	Allows the request to be redirected to an HTTP proxy server.
hostname	Name of the HTTP server.
ip-address	IP address of the HTTP server.
proxy_portnum	HTTP proxy server port number (1–65535).
username	Specifies the username to access the HTTP proxy server.
username	User login name.
password	Specifies the user password to access the HTTP proxy server.
password	Establishes password authentication.
running-config	Copies the current system configuration.
disk	Copies the current system configuration to a disk file.
filename	Name of the file to be created on disk.
startup-config	Copies the running configuration to the startup configuration (NVRAM).
tftp	Copies the running configuration to a file on a TFTP server.
hostname	Hostname of the TFTP server.

ip-address	IP address of the TFTP server.
remotefilename	Remote filename of the configuration file to be created on the TFTP server. Use the complete pathname.
startup-config	Copies the startup configuration.
disk	Copies the startup configuration to a disk file.
filename	Name of the startup configuration file to be copied to the local disk.
running-config	Copies the startup configuration to a running configuration.
tftp	Copies the startup configuration to a file on a TFTP server.
hostname	Hostname of the TFTP server.
ip-address	IP address of the TFTP server.
remotefilename	Remote filename of the startup configuration file to be created on the TFTP server. Use the complete pathname.
system-status disk	Copies the system status to a disk file.
filename	Name of the file to be created on the disk.
tech-support	Copies system information for technical support.
disk	Copies system information for technical support to a disk file.
filename	Name of the file to be created on disk.
tftp	Copies system information for technical support to a TFTP server.
hostname	Hostname of the TFTP server.
ip-address	IP address of the TFTP server.
remotefilename	Remote filename of the system information file to be created on the TFTP server. Use the complete pathname.
tftp	Copies an image from a TFTP server.
disk	Copies an image from a TFTP server to a disk file.
hostname	Hostname of the TFTP server.
ip-address	IP address of the TFTP server.
remotefilename	Name of the remote image file to be copied from the TFTP server. Use the complete pathname.
localfilename	Name of the image file to be created on the local disk.
running-config	Copies an image from a TFTP server to the running configuration.
hostname	Hostname of the TFTP server.
ip-address	IP address of the TFTP server.
remotefilename	Name of the remote image file to be copied from the TFTP server. Use the complete pathname.
startup-config	Copies an image from a TFTP server to the startup configuration.
hostname	Hostname of the TFTP server.
ip-address	IP address of the TFTP server.
remotefilename	Name of the remote image file to be copied from the TFTP server. Use the complete pathname.

Defaults

HTTP server port: 80

Default working directory for sysfs files: /local1

Command Modes EXEC

Usage Guidelines The **copy cdnfs** EXEC command copies data files out of the cdnfs to the sysfs for further processing. For example, you can use the **install** *imagefilename* EXEC command to provide the copied files to the command.

The **copy disk ftp** command copies files from a sysfs partition to an FTP server. The **copy disk startup-config** command copies a startup configuration file to NVRAM.

The **copy ftp disk** command copies a file from an FTP server to a sysfs partition.

Use the **copy ftp install** command to install an image file from an FTP server. Part of the image goes to the disk and part goes to the flash memory.

Use the **copy http install** command to install an image file from an HTTP server and install it on a local device. It transfers the image from an HTTP server to the Content Engine using HTTP as the transport protocol and installs the software on the device. Part of the image goes to the disk and part goes to the flash memory. You can also use this command to redirect your transfer to a different location or HTTP proxy server, by specifying the **proxy** *hostname* | *ip-address* option. A username and a password will have to be authenticated with the remote HTTP server if the server is password protected and requires authentication before the transfer of the software release file to the Content Engine is allowed.

Use the **copy running-config** command to copy the running system configuration to a sysfs partition, flash memory, or TFTP server. The **copy running-config startup-config** command is equivalent to the **write memory** command.

The **copy startup-config** command copies the startup configuration file to a TFTP server or to a sysfs partition.

The **copy system-status** command creates a file on a sysfs partition containing hardware and software status information.

The **copy tech-support tftp** command can copy technical support information to a TFTP server or to a a sysfs partition.

The **copy tftp disk** command copies a file from a TFTP server to a disk.

In Intel x86-based computers, such as the Content Engine, the basic input output system (BIOS) is the first software to execute when you power up or restart the system. The BIOS is responsible for initially configuring the processors, memory controller, RAM, and various other hardware devices. Additionally, the BIOS may perform power-on self test (POST) operations. After the BIOS completes the software initialization, the BIOS loads an operating system bootloader from the configured boot device. On ACNS devices, the BIOS typically loads the ACNS bootloader from the Content Engine flash memory (unless you are booting from a CD-ROM, which some models support). BIOS upgrades might be necessary to fix BIOS bugs related to hardware initialization. BIOS upgrades are less frequently required than typical operating system and application upgrades.

The Content Engine 7326 is the only hardware model that currently supports remote BIOS upgrades.

All BIOS files needed for a particular hardware model BIOS update are available on Cisco.com as a single .bin package file. This file is a special ACNS-installable .bin file that you can install by using the normal software update procedure.

To update the BIOS version on a Content Engine that supports BIOS version updates, you need the following items:

- FTP or HTTP server with the software files
- Network connectivity between the device to be updated and the server hosting the update files
- Appropriate .bin BIOS update file such as 7326_bios.bin

Caution Be careful when upgrading a flash BIOS. Make sure that the BIOS upgrade patch is the correct patch. If you apply the wrong patch, you can render the system unbootable, making it difficult or impossible to recover even by reapplying the proper patch.

Caution Because a failed flash BIOS update can have dire results, never update a flash BIOS without first connecting the system to an uninterruptable power supply (UPS).

To install the BIOS update file, use the **copy ftp install** or **copy http install** EXEC command as follows:

```
ContentEngine# copy ftp install ftp-server remote_file_dir 7326_bios.bin
```

or

```
ContentEngine# copy http install http-server remote_file_dir 7326_bios.bin
[portnumber]
```

After the BIOS update file is copied to your system, use the **reload** EXEC command to reboot as follows:

```
ContentEngine# reload
```

The new BIOS takes effect after the system reboots.

Examples

The following example copies an image file from an FTP server and installs the file on the local device:

```
CE-590# copy ftp install 10.1.1.1 //users2/ACNS400BR/boot ce590-ACNS-400.bin
Enter username for remote ftp server:biff
Enter password for remote ftp server:
Initiating FTP download...
printing one # per 1MB downloaded
Sending:USER biff
10.1.1.1 FTP server (Version) Mon Feb 28 10:30:36 EST
2000) ready.
Password required for biff.
Sending:PASS *****
User biff logged in.
Sending:TYPE I
Type set to I.
Sending:PASV
Entering Passive Mode (128,107,193,244,55,156)
Sending:CWD //users2/ACNS400BR/boot
CWD command successful.
Sending PASV
Entering Passive Mode (128,107,193,244,55,156)
Sending:RETR ce590-ACNS-400.bin
Opening BINARY mode data connection for ruby.bin (87376881 bytes).
################################################################################
```

```
writing flash component:
.............................................................
The new software will run after you reload.
CE-590#
```

The following example shows how to upgrade the BIOS. All output is written to a separate file (/local/local1/.bios_upgrade.txt) for traceability. The hardware-dependent files that are downloaded from Cisco.com for the BIOS upgrade are automatically deleted from the Content Engine after the BIOS upgrade procedure has been completed.

```
ce-7326# copy ftp install upgradeserver /bios/update53/derived/ 7326_bios.bin
Enter username for remote ftp server:myusername
Enter password for remote ftp server:
Initiating FTP download...
printing one # per 1MB downloaded
Sending:USER myusername
upgradeserver.cisco.com FTP server (Version wu-2.6.1-18) ready.
Password required for myusername.
Sending:PASS ********
Please read the file README_dotfiles
 it was last modified on Wed Feb 19 16:10:26 1997 - 2877 days ago
Please read the file README_first
 it was last modified on Wed Feb 19 16:05:29 1997 - 2877 days ago
User myusername logged in.
Sending:TYPE I
Type set to I.
Sending:PASV
Entering Passive Mode (128,107,193,240,57,37)
Sending:CWD /bios/update53/derived/
CWD command successful.
Sending PASV
Entering Passive Mode (128,107,193,240,146,117)
Sending:RETR 7326_bios.bin
Opening BINARY mode data connection for 7326_bios.bin (834689 bytes).
Fri Jan  7 15:29:07 UTC 2005
BIOS installer running!
Do not turnoff the system till BIOS installation is complete.
Flash chipset:Macronix 29LV320B
0055000.FLS:280000 [80000]
Erasing block 2f:280000 - 28ffff
Erasing block 30:290000 - 29ffff
Erasing block 31:2a0000 - 2affff
Erasing block 32:2b0000 - 2bffff
Erasing block 33:2c0000 - 2cffff
Erasing block 34:2d0000 - 2dffff
Erasing block 35:2e0000 - 2effff
Erasing block 36:2f0000 - 2fffff
Programming block 2f:280000 - 28ffff
Programming block 30:290000 - 29ffff
Programming block 31:2a0000 - 2affff
Programming block 32:2b0000 - 2bffff
Programming block 33:2c0000 - 2cffff
Programming block 34:2d0000 - 2dffff
Programming block 35:2e0000 - 2effff
Programming block 36:2f0000 - 2fffff
SCSIROM.BIN:260000 [20000]
Erasing block 2d:260000 - 26ffff
Erasing block 2e:270000 - 27ffff
Programming block 2d:260000 - 26ffff
Programming block 2e:270000 - 27ffff
PXEROM.BIN:250000 [10000]
Erasing block 2c:250000 - 25ffff
Programming block 2c:250000 - 25ffff
```

```
Primary BIOS flashed successfully
Cleanup BIOS related files that were downloaded....
The new software will run after you reload.
ce-7326#
```

Related Commands

install
reload
show running-config
show startup-config
write

cpfile

To make a copy of a file, use the **cpfile** EXEC command.

cpfile *oldfilename newfilename*

Syntax Description

oldfilename	Name of the file to copy.
newfilename	Name of the copy to be created.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to create a copy of a file. Only sysfs files can be copied.

Examples

The following example shows how to create a copy of a file:

```
ContentEngine# cpfile syslog.txt syslog.txt.save
```

Related Commands

copy
dir
lls
ls
mkfile
rename
rmdir

debug

To monitor and record caching application functions, use the **debug** EXEC command. To disable **debug**, use the **no** form of this command.

debug *option*

no debug *option*

Syntax Description

option	Specifies the debugger type; see the "Usage Guidelines" section for valid values.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Note We recommend that you use the **debug** command only at the direction of Cisco TAC because the Content Engine performance is affected when you enter the **debug** command. For more information, see the "Obtaining Technical Assistance" section on page 17.

You can use the **logging disk priority debug** global configuration command with the **debug** command. This configuration causes the debugging messages to be logged in the syslog file, which is available in the /local1 directory by default. You can then download the messages from the Content Engine, copy them to a local disk file (for example, using the **copy disk ftp** command), and forward the logs to Cisco TAC for further investigation. By default, system log messages are logged to the console and you need to copy and paste the output to a file. However, this method of obtaining logs is more prone to errors than capturing all messages in the syslog.txt file. When you use system logging to a disk file instead of system logging to a console, there is no immediate feedback that debug logging is occurring, except that the syslog.txt file gets larger (you can track the lines added to the syslog.txt file by entering the **type-tail syslog.txt follow** command). When you have completed downloading the system logs to a local disk, you must disable the debugging functions by using the **undebug** command (see the "undebug" section on page 2-819 for more details), and reset the level of logging disk priority to any other setting that you want (for example, **notice** priority).

Valid values for *option* are as follows:

aaa accounting	Debugs AAA accounting.
access-lists 300	Debugs the access control list.
dump	Dumps the access control list contents.
query	Queries the access control list configuration.

username *username*	Queries the access control list username.
groupname *groupnames*	Queries the access control list group name or names of groups of which the user is a member. Each group name must be separated by a comma.
acquirer	Debugs the acquirer.
error	Sets the debug level to error.
trace	Sets the debug level to trace.
all	Enables all debugging.
authentication	Debugs authentication.
http-request	Debugs the HTTP request authentication.
user	Debugs the user login against the system authentication.
authmod	Debugs the authentication module.
all	Displays the debug messages.
trace	Enables the request and response trace.
buf	Debugs the buffer manager.
all	Debugs all buffer manager functions.
dmbuf	Debugs the buffer manager dmbuf.
dmsg	Debugs the buffer manager dmsg.
cdnfs	Debugs the ACNS network file system (cdnfs).
cdp	Debugs Cisco Discovery Protocol (CDP).
adjacency	Debugs the CDP neighbor.
events	Debugs the CDP events.
ip	Debugs CDP IP.
packets	Debugs the packet-related CDP.
cli	Debugs the CLI command.
all	Debugs all CLI commands.
bin	Debugs the CLI command binary program.
parser	Debugs the CLI command parser.
cms	Debugs the CMS.

content-routing	Debugs the content routing.
all	Debugs all content routing.
ce	Debugs the Content Engine content routing.
config	Debugs the content routing configuration.
dns	Debugs the DNS content routing.
domain	Debugs the content routing domain.
keepalive	Debugs the content routing keepalive.
locks	Debugs the content routing locks.
lookup	Debugs the content routing lookup.
redir	Debugs the content routing redirection.
route	Debugs the content routing route.
rtsp	Debugs the RTSP content routing.
stats	Debugs the content routing statistics.
verbose	Debugs the content routing verbose mode.
dataserver	Debugs the data server.
all	Debuts all data server functions.
clientlib	Debugs the data server client library module.
server	Debugs the data server module.
dhcp	Debugs the DHCP.

distribution	Debugs the distribution components.
all	Debugs all distribution components.
error	Debugs all distribution components to error level 1 (show error).
trace	Debugs all distribution components to trace level 2 (show error and trace).
metadata-receiver	Debugs the metadata receiver distribution component.
error	Debugs the metadata receiver distribution component to error level 1.
trace	Debugs the metadata receiver distribution component to trace level 2.
metadata-sender	Debugs the metadata sender distribution component.
error	Debugs the metadata sender distribution component to error level 1.
trace	Debugs the metadata sender distribution component to trace level 2.
mcast-receiver	Debugs the multicast receiver distribution component.
error	Debugs the multicast receiver distribution component to error level 1.
trace	Debugs the multicast receiver distribution component to trace level 2.
mcast-sender	Debugs the multicast sender distribution component.
error	Debugs the multicast sender distribution component to error level 1.
trace	Debugs the multicast sender distribution component to trace level 2.
unicast-data-receiver	Debugs the unicast receiver distribution component.
error	Debugs the unicast receiver distribution component to error level 1.
trace	Debugs the unicast receiver distribution component to trace level 2.
unicast-data-sender	Debugs the unicast sender distribution component.
error	Debugs the unicast sender distribution component to error level 1.
trace	Debugs the unicast sender distribution component to trace level 2.

dns	Debugs the DNS.
all	Debugs all of the DNS.
cache	Debugs the DNS cache.
client	Debugs the DNS client.
config	Debugs the DNS configuration.
driver	Debugs the DNS driver.
memory	Debugs the DNS memory.
parser	Debugs the DNS parser.
response	Debugs the DNS response.
retry	Debugs the DNS response.
servers	Debugs the DNS servers.
emdb	Debugs the embedded database.
level	(Optional) Debug level.
(0-16)	Debug level 0 through 16.
ftp-native	Debugs the native FTP functions (including fetching and caching files from an FTP server, posting files to an FTP server, and performing directory listings on an FTP server).
all	Debugs all native FTP functions.
cache	Debugs the cache proxy that is used for native FTP caching (the cache proxy resides on the Content Engine that is operating in nontransparent proxy mode to support the native FTP requests).
client	Debugs the native FTP client.
control-proxy	Debugs the control proxy that is used for native FTP caching (the control proxy resides on the Content Engine that is operating in nontransparent proxy mode to support the native FTP requests).
dns	Debugs the control proxy that is used for DNS resolution of native FTP requests.
parser	Debugs the parser that is used for the native FTP caching.
proxy-comm	Debugs the proxy communications that are used for the native FTP functions.
server	Debugs the native FTP server.
ftp-over-http	Debugs the FTP functions for the FTP-over-HTTP requests (including fetching and caching files from an FTP server).
all	Debugs all FTP functions for the FTP-over-HTTP requests.
cache	Debugs the FTP cache (the Content Engine that is operating in nontransparent proxy mode to cache the contents of the FTP-over-HTTP requests).
client	Debugs the FTP client (end users who are issuing an FTP-over-HTTP request from their browsers).
server	Debugs the FTP server (for the FTP-over-HTTP requests).

http	Debugs the HTTP commands.
all	Debugs all HTTP functions.
cache	Debugs the HTTP cache.
content-router	Debugs the HTTP content routing.
header	Debugs an HTTP header.
hit	Debugs an HTTP hit.
miss	Debugs an HTTP miss.
pac-file-server	Debugs HTTP for the dynamic proxy autoconfiguration feature.
parser	Debugs the HTTP parser.
plugin	Debugs the HTTP plug-in.
proxy	Debugs the HTTP proxy.
server	Debugs the HTTP server.
http-authcache	Debugs the authentication cache.
all	Debugs all the authentication cache functions.
application	Debugs the application module.
cli	Debugs the CLI module.
daemon	Debugs the daemon client module.
https	Debugs HTTPS.
all	Debugs all HTTPS functions.
cli	Debugs the HTTPS CLI.
header	Debugs the HTTPS header.
parser	Debugs the HTTPS parser.
proxy	Debugs the HTTPS proxy.
icap	Debugs ICAP.
client	Debugs the ICAP client (caching proxy) processing.
daemon	Debugs the ICAP daemon processing.
icp	Debugs ICP.
all	Debugs all ICP functions.
client	Debugs the ICP client module.
ex	Debugs the ICP exclude module.
heal	Debugs the ICP healing module.
main	Debugs the ICP main module.
parse	Debugs the ICP parser module.
print	Debugs the ICP printer module.
server	Debugs the ICP server module.
utils	Debugs the ICP utilities module.

iptv program-manager	Debugs IP/TV Program Manager.
admin	Debugs the administration module in IP/TV Program Manager.
all	Debugs all modules in IP/TV Program Manager.
cdm-api	Debugs the Content Distribution Manager APIs in IP/TV Program Manager.
db	Debugs the IP/TV Program Manager database.
ftpd	Debugs the FTP daemon in IP/TV Program Manager.
gui-engine	Debugs the GUI engine in IP/TV Program Manager.
guide	Debugs the program guide in IP/TV Program Manager.
live-manager	Debugs the live manager in IP/TV Program Manager.
on-demand-manager	Debugs the On Demand Manager in IP/TV Program Manager.
publisher	Debugs the program publisher in IP/TV Program Manager.
question-manager	Debugs Question Manager in IP/TV Program Manager.
session-description	Debugs the session description module in IP/TV Program Manager.
logging	Debugs logging.
all	Debugs all logging functions.
ntp	Debugs NTP.
pac-file-server	Debugs the dynamic proxy autoconfiguration feature.
all	(Optional) Debugs all PAC file server features.
config	(Optional) Debugs the PAC file server configuration.
locks	(Optional) Debugs the PAC file server locks.
lookup	(Optional) Debugs the PAC file server lookup.
route	(Optional) Debugs the PAC file server routes.
verbose	(Optional) Enables the verbose debugging messages for the PAC file server.
pre-load	Debugs the preload.
all	(Optional) Debugs all preload functions.
rbcp	Debugs the RBCP (Router Blade Configuration Protocol) functions.
rpc	Displays the remote procedure calls (RPC) logs.
detail	Displays the RPC logs of priority “detail” level or higher.
trace	Displays the RPC logs of priority “trace” level or higher.

rtsp	Debugs the RTSP functions.
gateway	Debugs the RTSP gateway.
error	Debugs the RTSP gateway to level 1 (show error).
trace	Debugs the RTSP gateway to level 2 (show error and trace).
proxy media-real	Debugs RTSP RealProxy.
real-all	Debugs all RealProxy plug-ins.
real-allowance	Debugs RealProxy allowance plug-in.
real-cache	Debugs RealProxy cache plug-in.
real-stats	Debugs RealProxy statistics plug-in.
rule	Debugs the Rules Template.
action	Debugs the rule action.
all	Debugs all rule functions.
pattern	Debugs the rule pattern.
snmp	Debugs SNMP.
all	Debugs all SNMP functions.
cli	Debugs the SNMP CLI.
main	Debugs the SNMP main.
mib	Debugs the SNMP MIB.
traps	Debugs the SNMP traps.
standby	Debugs standby.
all	(Optional) Debugs all standby functions.
stats	Debugs the statistics.
all	Debugs all statistics functions.
collection	Debugs the statistics collection.
computation	Debugs the statistics computation.
history	Debugs the statistics history.
translog	Debugs the transaction logging.
all	Debugs all transaction logging.
archive	Debugs the transaction log archive.
export	Debugs the transaction log FTP export.
tvout	Debugs the TV output.
all	Debugs all TV output.
device	Debugs the TV output device.
playlist	Debugs the TV output playlist.
schedule	Debugs the TV output schedule.

url-filter	Debugs the URL filtering.
local-list	Debugs the URL local bad or local good list filtering.
N2H2	Debugs the URL N2H2 filtering.
websense	Debugs the URL Websense filtering.
wccp	Debugs the WCCP information.
all	Debugs all WCCP functions.
detail	Debugs the WCCP details.
error	Debugs the WCCP errors.
events	Debugs the WCCP events.
keepalive	Debugs the WCCP keepalives that are sent to the applications.
packets	Debugs the WCCP packet-related information.
slowstart	Debugs the WCCP slow start.
wi	Debugs the web interface.
wmt	Debugs the WMT component.
error	Debugs the WMT level 1 functionality. For more information, see the "Using WMT Error Logging" section on page 2-136.
client-ip *cl-ip-address*	(Optional) Debugs the request from a specific client IP address to level 1 (show error).
server-ip *sv-ip-address*	(Optional) Debugs the request to a specific server IP address to level 1 (show error).
trace	Debugs the WMT level 2 functionality.
client-ip *cl-ip-address*	Debugs the request from a specific client IP address to level 2 (show error and trace).
server-ip *sv-ip-address*	Debugs the request to a specific server IP address to level 2 (show error and trace).

Debugging Cdnfs

You can use the **debug cdnfs** command to monitor the lookup and serving of pre-positioned files. If pre-positioned files are available in cdnfs but are not served properly, you can use cdnfs debug.

Using WMT Error Logging

In the ACNS 5.2 software, WMT error logging was enhanced. More information is now logged about the following events:

- When a WMT client is abruptly disconnected
- When any WMT streams are cleared on the Content Engine

Error logs are in the same format and location as syslogs. The WMT log messages are logged to /local1/errorlog/wmt_errorlog.current.

You can configure the Content Engine for WMT error logging by using the **debug wmt error** EXEC command. This command debugs WMT level 1 functionality.

Logging WMT Client Disconnects

When a WMT client is disconnected abruptly, the reasons for the client disconnect (for example, the request was blocked by the rules, the maximum incoming or outgoing bit-rate limit was reached, the maximum incoming or outgoing bandwidth limit was reached) are logged in ACNS software error logs.

The client information includes the client IP address, the server IP address, the requested URL, the client protocol, the version of the client media player, the number of packets that the client received, and the number of packets that the server sent.

Related Commands

logging
show debugging
undebug

delfile

To delete a file, use the **delfile** EXEC command.

delfile *filename*

Syntax Description

filename	Name of the file to delete.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to remove a file from a sysfs partition.

Examples

The following example shows how to delete a file:

```
ContentEngine# delfile /local1/tempfile
```

Related Commands

cpfile
deltree
mkdir
mkfile
rmdir

deltree

To remove a directory with its subdirectories and files, use the **deltree** EXEC command.

deltree *directory*

Syntax Description		
	directory	Name of the directory tree to delete.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to remove a directory and all files within the directory from the Content Engine sysfs file system. Do not remove files or directories required for proper Content Engine functioning.

Examples

The following example shows how to delete a directory from the /local1 directory:

```
ContentEngine# deltree /local1/testdir
```

Related Commands

delfile
mkdir
mkfile
rmdir

device

To configure the mode of operation on a device as a Content Distribution Manager, Content Engine, Content Router, or IP/TV Program Manager, use the **device** global configuration command. To reset the mode of operation on a device, use the **no** form of this command.

device mode {**content-distribution-manager | content-engine | content-router | program-manager**}

no device mode {**content-distribution-manager | content-engine | content-router | program-manager**}

Syntax Description

mode	Sets the mode of operation of a device to Content Distribution Manager, Content Engine, Content Router, or IP/TV Program Manager.
content-distribution-manager	Configures the device operation mode as a Content Distribution Manager.
content-engine	Configures the device operation mode as a Content Engine.
content-router	Configures the device operation mode as a Content Router.
program-manager	Configures the device operation mode as an IP/TV Program Manager.

Defaults

The default device operation mode is Content Engine.

Command Modes

global configuration

Usage Guidelines

A Content Distribution Manager is the content management and device management station of an ACNS network that allows you to specify what content is to be distributed, and where the content should be distributed. If a Content Router is deployed in the ACNS network, this device directs requests from a client to the nearest Content Engine for content delivery. A Content Engine is the device that serves content to the clients. There are typically many Content Engines deployed in an ACNS network, each serving a local set of clients. IP/TV brings movie-quality video over enterprise networks to the desktop of the ACNS network user.

IP/TV Program Manager is a network application that lets you manage both scheduled and on-demand video content for your ACNS network. If the device is configured as IP/TV Program Manager, you can distribute IP/TV content using manifest files (where the origin server and content to be distributed are specified) and channels (which specify which devices are to receive the content) to the ACNS network. For more information regarding the deployment of these devices in an ACNS network, see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments*.

Because different device modes require disk space to be used in different ways, disk space must also be configured when the device mode changes from being a Content Engine or Content Router or Content Distribution Manager to an IP/TV Program Manager (or the other way around). You must reboot the device before the configuration changes to the device mode take effect.

Disks must be configured before device configuration is changed. Use the **disk configure** command to configure the disk before reconfiguring the device to the Content Engine or Content Router mode. Disk configuration changes using the **disk configure** command takes effect after the next device reboot.

Caution Be careful when you use the **disk cancel-config** command to store data on the device before the device is reconfigured because the data will be lost after the next reboot.

Caution Be careful when you use the **disk config** command because this command deletes all existing sysfs, mediafs, and cfs content when the disk configuration takes effect during reboot. The content in the cdnfs, however, is preserved.

To enable ACNS network-related applications and services, use the **cms enable** command. Use the **no** form of this command to disable the ACNS network.

All ACNS devices ship from the factory as Content Engines. Before configuring network settings for Content Distribution Managers and Content Routers using the CLI, you must change the device from a Content Engine to the proper device mode.

Configuring the device mode is not a supported option on all hardware models. However, you can configure some hardware models to operate as any one of the four content networking device types. Devices that can be reconfigured using the **device mode** global configuration command are shipped from the factory by default as Content Engines.

The following hardware models support the device mode configuration:

- CE-7305
- CE-7326
- CE-565
- CE-566

To change the device mode of your Content Engine, you must also configure the disk space allocations, as required by the different device modes, and reboot the device for the new configuration to take effect.

When you change the device mode of a Content Engine to a Content Router or Content Distribution Manager, you may need to reconfigure the system file system (sysfs). However, Content Routers and Content Distribution Managers do not require any disk space other than sysfs. When you change the device mode to a Content Router or a Content Distribution Manager, disk configuration changes are not required because the device already has some space allotted for sysfs. sysfs disk space is always preconfigured on a factory-fresh ACNS network device. See the "Disk Space-Allocation Guidelines for Content Routers" section on page 2-148 and "Disk Space-Allocation Guidelines for Content Distribution Managers" section on page 2-148 for more information.

If you are changing the device mode of a Content Router or a Content Distribution Manager back to a Content Engine, you must configure disk space allocations for the caching (cfs), pre-positioning (cdnfs), streaming (mediafs), and system use (sysfs) file systems that are used on the Content Engine. You can configure disk space allocations either before or after you change the device mode to a Content Engine.

Examples

The following examples show the configuration from the default mode, Content Engine, to the Content Distribution Manager, Content Router, and Content Engine modes:

```
ContentEngine(config)# device mode content-distribution-manager

CDM(config)# device mode content-router

ContentRouter(config)# device mode content-engine
```

Related Commands

show device-mode

dir

To view a long list of files in a directory, use the **dir** EXEC command.

dir [*directory*]

Syntax Description

directory	(Optional) Name of the directory to list.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to view a detailed list of files contained within the working directory, including names, sizes, and time created. The equivalent command is **lls**.

Examples

The following example shows how to view a list of files in a directory:

```
ContentEngine# dir
size            time of last change                name
--------------  -------------------------          -----------
       3931934  Tue Sep 19 10:41:32 2000           errlog-cache-20000918-164015
           431  Mon Sep 18 16:57:40 2000           ii.cfg
           431  Mon Sep 18 17:27:46 2000           ii4.cfg
           431  Mon Sep 18 16:54:50 2000           iii.cfg
          1453  Tue Sep 19 10:34:03 2000           syslog.txt
          1024  Tue Sep 19 10:41:31 2000  <DIR>    testdir
```

Related Commands

lls
ls

disable

To turn off privileged EXEC commands, use the **disable** EXEC command.

disable

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

The **disable** command places you in the user-level EXEC shell. To turn privileged EXEC mode back on, use the **enable** command.

Examples

The following example shows how to enter the user-level EXEC mode:

```
ContentEngine# disable
ContentEngine>
```

Related Commands

enable

disk (EXEC)

To configure disks and allocate disk space for devices that are using the ACNS software, use the **disk** EXEC command.

disk add *diskname* {**cdnfs** {**remaining** | *disk-space*} [[**cfs** | **mediafs** | **sysfs**] {**remaining** | *disk-space*}] | **cfs** {**remaining** | *disk-space*} [[**cdnfs** | **mediafs** | **sysfs**] {**remaining** | *disk-space*}] | **mediafs** {**remaining** | *disk-space*} [[**cdnfs** | **cfs** | **sysfs**] {**remaining** | *disk-space*}] | **sysfs** {**remaining** | *disk-space*} [[**cdnfs** | **cfs** | **mediafs**] {**remaining** | *disk-space*}]}

disk cancel-config

disk config sysfs {**remaining** | *disk-space*} [**cdnfs** {**remaining** | *disk-space*}] [**cfs** {**remaining** | *disk-space*}] [**mediafs** {**from-unused-cdnfs** | **remaining** | *disk-space*}]

disk delete-partitions *diskname*

disk mark *diskname* {**bad** | **good**}

disk recover-disk00

disk reformat *diskname*

disk scan-errors *diskname*

disk unuse *diskname*

Syntax Description

add	Specifies the addition of a single disk with specified partitions.
diskname	Name of the disk to be added (disk01, disk02, and so on).
cdnfs	Specifies a file system used for pre-positioned files used in the ACNS network. **Note** The ACNS network was formerly known as CDN.
remaining	Specifies the remaining disk size after other file system disk sizes have been specified. Only one file system can be used with this keyword.
disk-space	Size of the disk partition, using an integer followed by MB, GB, or %, to indicate the size in megabytes, gigabytes, or as a percentage of the total system storage.
cfs	Specifies the file system used for demand-cached HTTP and FTP content.
mediafs	Specifies the file system used for demand-cached RealMedia and WMT streaming content.
sysfs	Specifies the file system used for log files, user-downloaded configuration or image files, core files, and SmartFilter files. The minimum required *disk-space* is 1 GB.
cancel-config	Cancels the disk configuration change (prior to reboot only).
config	Configures the disk space.
from-unused-cdnfs	Configures the disk space that is not used for the cdnfs.
delete-partitions	Deletes all disk partitions on the specified disk drive.

diskname	Name of the disk to be deleted (disk01, disk02, and so on).
mark	Marks a disk drive as good or bad.
diskname	Name of the disk to be marked (disk01, disk02, and so on).
bad	Marks the disk drive as bad.
good	Marks the disk drive as good.
recover-disk00	Recovers the system disk (disk00).
reformat	Performs a low-level format of the SCSI, IDE, or SATA disks and remaps disk errors.
diskname	Name of the disk to be reformatted (disk01, disk02, and so on).
scan-errors	Scans SCSI, IDE, or SATA disks for errors and remaps the bad sectors, if they are unused.
diskname	Name of the disk to be scanned for errors (disk01, disk02, and so on).
unuse	Stops applications from using a disk drive.
diskname	Name of the disk to be stopped for application use (disk01, disk02, and so on).

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

The disk space in the ACNS software is allocated on a per-file system basis, rather than on a per-disk basis. You can configure your overall disk storage allocations according to the kinds of client protocols that you expect to use and the amount of storage that you need to provide for each of the functions. Use the **disk add** EXEC command to add a single disk with the specified partitions.

Note For details on the Cisco ACNS software disk storage for Content Engines, see the *Cisco ACNS Software Upgrade and Maintenance Guide*.

In the ACNS 5.2.x software and earlier releases, the cndfs, mediafs, and sysfs partitions use the ext2 file system. With ext2 file systems, if the system crashed or if the system is not shut down cleanly, a file system check of these partitions takes a long time. If there are sector failures on the disk, the time to perform a file system check with an ext2 file system increases even more. In the ACNS 5.3 software release and later releases, the ext3 file system is used instead of the ext2 file system. By migrating to the ext3 file system, the amount of time required to perform a file system check of the cndfs, mediafs, and sysfs partitions is decreased, which increases the availability of the Content Engine. If you are upgrading from an earlier release of the ACNS software, the ext2 file system is automatically converted to the ext3 file system when you upgrade to the ACNS 5.3 software and later releases.

Use the **disk config** command to configure disk allocations. This command takes the file system type and size as parameters. You can designate the size in megabytes, gigabytes, or as a percentage of the system total storage, or you can enter the **remaining** option to use the remaining available disk space. For mediafs, you can enter the **from-unused-cdnfs** option to give the cdnfs the bulk of the storage space and the remaining unused cdnfs storage space to mediafs.

Note For information on disk configuration requirements, see the *Cisco ACNS Software Upgrade and Maintenance Guide*.

When you are using the cfs and the mediafs, the content is loaded on demand and the older content is automatically removed to make room for the newer content. For this reason, you can obtain reasonably good performance using relatively small amounts of disk space. However, if the cdnfs disk space is exhausted, the new content cannot be acquired or distributed. You must manually remove the existing content or use the **disk config** command to increase the cdnfs disk space. To maximize the availability of the cdnfs disk space, use the **cdnfs remaining** option with the **mediafs from-unused cdnfs** option.

Caution Be careful when using the **disk config** command because this command deletes all existing sysfs, mediafs, and cfs content when the disk configuration takes effect during reboot. The content in the cdnfs, however, is preserved.

When you are using a Content Engine in an edge or branch office environment, and in the absence of other more specific requirements, use the following disk configuration as a default configuration:

```
Content Engine# disk config sysfs 5% cfs 25% cdnfs remaining
```

where 5 percent of the total storage is allocated to the sysfs, 25 percent is allocated to the cfs, and the remaining disk space is allocated to the cdnfs.

The cdnfs and mediafs amounts are reported by the actual usable amounts of storage for applications. Due to the internal file system overhead of approximately 3 percent, the reported amounts may be smaller than what you configured. The cdnfs space is allocated with a higher priority than mediafs, so if you configured mediafs and cdnfs, mediafs will be reduced by the amount of the total cdnfs and mediafs overhead. If you have not configured mediafs, cdnfs will be reduced by the amount of the overhead.

The disk configuration does not take effect until after the next reboot. To view the configuration after the next reboot, use the **show disks configured** command.

To view disk details, use the **show disks details**.

Note The **show disks details** command shows the amount of disk space that is allocated to system use. The CE-7325 and CE-7305 each use 10.5 GB, the CE-565 uses 8.2 GB, and the CE-510 uses 6 GB. On earlier versions of the devices, the system usage space is 3 to 4 GB. This detail is not shown by using the **show disks current** command.

To show the space allocation in each individual file system type, use the **show statistics cdnfs**, **show statistics cfs**, or **show statistics mediafs** commands.

Note For information on disk allocation guidelines for Content Engines, see the *Cisco ACNS Software Upgrade and Maintenance Guide*.

Note You should configure the mediafs storage space only if RealProxy or WMT files are being cached.

For higher-end models, such as the CE-7320 that might be used as a dedicated HTTP cache or RealProxy cache, you could give either cfs storage or mediafs storage more disk space.

Note You must configure the mediafs storage and enable the RealProxy Real-Time Streaming Protocol (RTSP) proxy service before any RealProxy files can be cached in the mediafs storage space.

After upgrading from an earlier release of ACNS software to the ACNS 5.3 software release and later releases, your disk space allocation remains the same as previously configured. If you want to configure the mediafs to use unused cdnfs storage space, you must configure this option either through the CLI or through the Content Distribution Manager GUI and then reload the software for the change to take effect.

Disk Space-Allocation Guidelines for Content Routers

In the ACNS 5.x software, Content Routers are used as DNS servers for the delegated DNS zone used in simplified hybrid routing. The DNS servers do not store any content and do not participate in acquisition or distribution of the pre-positioned content. The only disk space that needs to be configured on the Content Router is the sysfs.

Disk Space-Allocation Guidelines for Content Distribution Managers

Content Distribution Managers are used to manage content distribution for ACNS networks. Because the Content Distribution Manager does not store the content, the only file system that needs to be configured is the sysfs.

To cancel the disk configuration, use the **disk cancel-config** command.

Note The **disk cancel-config** command is effective only before a reboot. After reboot, the allocation has already taken effect and can be changed only by entering another **disk config** command.

Use the **disk add** command to add a single disk with specified partitions.

Use the **disk raid-array add-array** command to create a logical disk for the Storage Array that is recognized by the CDM-4650 RAID controller.

Use the **disk raid-array repair** command to rebuild a RAID disk array after a single disk in the array fails.

Note In the ACNS 5.x software, the **disk add** command does not support disk00 but supports disk01 or higher versions, where the drive in the slot is a blank new replacement disk. Use the **disk recover-disk00** command rather than the **disk add** command to add disk00.

Remapping of Bad Sectors on Disk Drives

The ACNS 5.3 software release and later releases extend the support for remapping unused bad disk sectors to IDE and SATA disk drives, as well as SCSI disk drives.

The **disk scan-errors** *diskname* EXEC command scans SCSI, IDE, or SATA disks for errors and remaps the bad sectors if they are unused. This command also reports the failed and fixed or unfixed sectors that are found. Use this command to determine whether or not the disk drive has any errors.

The **disk reformat** *diskname* EXEC command performs a low-level format of the SCSI, IDE, or SATA disks. Use this command only as a final resort when the **disk scan-errors** command has been unable to fix all of the disk drive errors. This command erases all of the content on the disk.

If a disk drive continues to report a failure after you have used both the **disk scan-errors** and **disk reformat** commands, you must replace the disk drive.

Caution Be careful when using the **disk reformat** *diskname* command because this command causes all content on the specified disk to be deleted. You should use the **disk reformat** command only if you are unable to remap the bad sectors using the **disk scan-errors** command.

In the ACNS 5.2 software release and later releases, support for reformatting a SCSI drive was added. In the ACNS 5.3 software release, this capability is extended to include IDE and Serial Advanced Technology Attachment (SATA) drives.

Removing All Disk Partitions on a Single Disk Drive

Use the **disk delete-partitions** EXEC command to remove all disk partitions on a single disk drive.

Caution The **disk delete-partitions** EXEC command will erase everything on the specified disk.

Typically, this command is used when you want to add a new disk drive that was previously used with another operating system (for example, a Microsoft Windows or Linux operating system). When asked if you want to erase everything on the disk, specify "yes" to proceed.

Manually Marking and Unmarking Content Engine Disk Drives

A disk drive that has been previously marked bad on a Content Engine will remain in the "Not used state" until you manually unmark it as follows:

- Use the **disk add** EXEC command on a Content Engine. The "Not used state" is reset if you use the **disk add** command.
- Use the **disk mark** EXEC command to mark one or all disk drives manually as good (being used) or bad (will not be used after a reload).

Note For information on the procedure to manually mark and unmark Content Engine disk drives, see Chapter 21 of the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments*.

Note The ACNS software automatically marks a drive as bad in certain situations, such as when the threshold for disk drive errors for a particular disk drive is reached. To change the default threshold, use the **disk error-handling threshold** global configuration command. Specify **0** if you never want the disk drive to be marked as bad.

Stopping Applications from Using a Disk Drive

In the ACNS 5.3 software release, the **disk unuse** EXEC command was added. This command allows you to stop applications from using a specific disk drive (for example, disk01) without having to reboot the device:

```
ContentEngine# disk unuse disk01
```

The disk unuse feature cannot be used with disk00 (the first disk drive) or with the drive that contains the /local/local1 directory (for example, if disk01 contains the /local/local1 directory, then you cannot use the **disk unuse** command with disk01). For more information, see the *Cisco ACNS Software Update and Maintenance Guide.*

This command stops and restarts all applications that are currently using the specified disk drive (for example, disk02 [/local/local2] or disk03), unmounts, and deletes all the partitions on the specified disk. Using this command unmounts all file systems, including cfs, cdnfs, and mediafs.

Note For information on disk media file system issues when downgrading to the ACNS 5.0 software, see the *Cisco ACNS Software Upgrade and Maintenance Guide*.

Examples

The following example is appropriate for a Content Engine in a network edge or branch office environment that does both HTTP caching and ACNS network distribution:

```
ContentEngine# disk config sysfs 5% cfs 25% cdnfs remaining
```

The above example and the following example assume at least 20 GB of disk storage, because a minimum of 1 GB is required for the sysfs.

The following example includes the disk space allocation for proxying RealMedia or WMT content, 5 percent of the total storage is allocated to the sysfs and 25 percent is allocated to the cfs. The remaining 70 percent is allocated to the cdnfs and the mediafs; the mediafs uses only the disk space that remains unused by the cdnfs.

```
ContentEngine# disk config sysfs 5% cfs 25% cdnfs remaining mediafs from-unused-cdnfs
```

The following example allocates 10 percent for the sysfs to allow more space for transaction log files and is for an environment where only HTTP caching is required.

Note If you enable this feature, each HTTP request generates a line in the transaction log.

```
ContentEngine# disk config sysfs 10% cfs remaining
```

The following examples show usage of the **disk unuse** command and the resultant actions:

```
ce# disk unuse disk00
Disk00 can not be unuseed!

ce# disk unuse disk01
Disk01 has mounted SYSFS and can not be unused!

ce# disk unuse disk02
This will restart applications currently using disk02 and unmount all partitions on
disk02.
Do you want to continue? (yes/no) [no]no
Disk02 not unused.

ce# disk unuse disk02
This will restart applications currently using disk02 and unmount all partitions on
disk02.
Do you want to continue? (yes/no) [no]yes
Disk02 has been unused. No application is using disk02 now.

ce# disk unuse disk02 delete-partitions
This will restart applications currently using disk02 and unmount and *delete* all
partitions on disk02.
Do you want to continue? (yes/no) [no]yes
Disk02 has been unused. No application is using disk02 now.
And all partitions on disk02 are deleted.
```

The following example shows how to display the current disk space configuration:

```
ContentEngine# show disks current
Local disks:
  SYSFS                 1.0GB       7.2%
  CFS                   0.0GB       0.0%
  MEDIAFS               0.0GB       0.0% (from-unused-cdnfs)
  CDNFS                 12.4GB  89.8%
  FREE                 0.0GB     0.1%
Network-attached disks:
  NONE
```

The following example shows how to view the configuration after a reboot:

```
ContentEngine# show disks configured
SYSFS                 5%
CFS                   25%
MEDIAFS               0%
CDNFS                 remaining
```

The following example shows how to view disk details:

```
ContentEngine# show disks details
disk00: Normal          (h00 c00 i00 l00 - Int DAS-SCSI)   17499MB( 17.1GB)
        disk00/04: PHYS-FS       12730MB( 12.4GB) mounted internally
        disk00/04: CDNFS         12730MB( 12.4GB) mounted internally
        disk00/05: SYSFS          1023MB(  1.0GB) mounted at /local1
        System use:               3316MB(  3.2GB)
        FREE:                       16MB(  0.0GB)
disk01: Not present or not responding
No NAS share is attached to this device.
ContentEngine#
```

The following examples show how to view space allocation in each file system type:

```
ContentEngine# show statistics cdnfs

CDNFS Statistics:
------------------
Volume on disk06/00:
  size of physical file system:               5078640 KB
  space assigned for CDNFS purposes:         1048576 KB
  number of CDNFS entries:                         2 entries
  space reserved for CDNFS entries:               25 KB
  available space for new entries:           1048551 KB
  ACNS 4.x legacy ECDN files (total):              0 KB
  ACNS 4.x legacy ECDN files (unused):             0 KB
  physical file system space in use:            33992 KB
  physical file system space free:            5044648 KB
  physical file system percentage in use:           1 %

ContentEngine# show statistics cfs
CFS statistics
--------------
Volume 0
          Total disk space             = 1070596096
          Total disk space used        = 6291456
          Total disk objects read      = 0
          Total disk objects write     = 0
          Total bytes of disk read     = 0
          Total bytes of disk write    = 0
          Disk read errors             = 0
          Disk write errors            = 0
```

```
ContentEngine# show statistics mediafs
MEDIAFS Statistics:
------------------
Volume on disk00/04:
  size of physical file system:              13316616 KB
  space assigned for MEDIAFS purposes:      12268040 KB
  Space used by Real Proxy:                    72336 KB
  Space used by WMT Proxy:                      2524 KB
  physical file system space in use:          125580 KB
  physical file system space free:          13191036 KB
  physical file system percentage in use:           1 %

Volume on disk01/00:
  size of physical file system:              15312740 KB
  space assigned for MEDIAFS purposes:      15312740 KB
  Space used by Real Proxy:                    45608 KB
  Space used by WMT Proxy:                         0 KB
  physical file system space in use:           79568 KB
  physical file system space free:          15233172 KB
  physical file system percentage in use:           1 %

ContentEngine#
```

The following examples show how to allocate disk space:

```
ContentEngine# disk config sysfs 10% cfs 80% mediafs 10% cdnfs 0%

ContentEngine# disk config sysfs 10% cfs 10% mediafs 80% cdnfs 0%
```

The first example might depict an ISP deployment or enterprise data center deployment that serves a large number of users per Content Engine (over 200), where the cfs storage space should be higher. In these types of deployments, the additional cfs storage space helps improve HTTP caching. Where the Content Engine is not deployed as part of an ACNS network, you do not need to configure any cdnfs storage.

If you want to use both RealProxy or WMT caching and HTTP caching, the following example shows how you could evenly split the disk space between cfs storage and mediafs storage:

```
ContentEngine# disk config sysfs 10% cfs 45% mediafs 45% cdnfs 0%
```

The following example shows how to configure the sysfs:

```
CDM4630# disk config sysfs 5GB
```

The following example shows how to cancel disk configuration:

```
ContentEngine# disk cancel-config
Disk configuration canceled successfully
```

The following example shows how to delete disk partitions:

```
ContentEngine# disk delete-partitions disk03
This will erase everything on disk. Are you sure? [no] yes
```

Related Commands

disk (global configuration mode)
show cdnfs
show cfs
show disks
show disks details
show mediafs
show statistics

disk (global configuration)

To configure how disk errors should be handled and to define a disk device error-handling threshold, use the **disk** global configuration command. To remove the device error-handling options, use the **no** form of this command.

disk error-handling {**reload** | **remap** | **threshold** *number*}

no disk error-handling {**reload** | **remap** | **threshold** *number*}

Syntax Description

error-handling	Configures disk error handling.
reload	Reloads the disk if the system file system (sysfs) (disk00) has problems.
remap	Sets the disk to attempt to remap disk errors automatically.
threshold	Sets the number of disk errors allowed before the disk is marked as bad.
number	Number of disk errors allowed before the disk is marked as bad (0–100). The default is 1. The value 0 means that the disk should never be marked as bad.

Defaults

error-handling threshold *number*: 10

Command Modes

global configuration

Usage Guidelines

In order to operate properly, the Content Engine must have the following critical disk drives:

- The first disk drive that is referred to as disk00.
- The disk drive that contains the first sysfs (system file system) partition.

  The sysfs partition is used to store log files, including transaction logs, system logs (syslogs), and internal debugging logs. It can also be used to store image files and configuration files on a Content Engine.

Note A critical drive is a disk drive that is either disk00 or a disk drive that contains the first sysfs partition. Smaller single disk drive Content Engines have only one critical disk drive. Higher-end Content Engines that have more than one disk drive may have more than one critical disk drive.

When a Content Engine is booted and a critical disk drive is not detected at system startup time, the ACNS system on the Content Engine runs at a degraded state. If one of the critical disk drives goes bad at run time, the ACNS system applications can malfunction, hang, or crash, or the ACNS system can hang or crash. You must monitor the critical disk drives on a Content Engine and report any disk drive errors to Cisco TAC.

With an ACNS system, a disk device error is defined as any of the following events:

- A Small Computer Systems Interface (SCSI) or Integrated Drive Electronics (IDE) device error is printed by a Linux kernel.
- A disk device access by an application (for example, an open(2), read(2), or write(2) system call) fails with an EIO error code.
- A disk device that existed at startup time is not accessible at run time.

In the ACNS 5.2 software and later releases, you can monitor Content Engine disk drives. The disk status is recorded in flash (nonvolatile storage). When an error on a Content Engine disk device occurs, a message is written to the system log (syslog) if the sysfs partition is still intact, and an SNMP trap is generated if SNMP is configured on the Content Engine.

In addition to tracking the state of critical disk drives, you can define a disk device error-handling threshold on the Content Engine. If the number of disk device errors reaches the specified threshold, the corresponding disk device is automatically marked as bad. The ACNS system does not stop using the bad disk device immediately; it stops using the bad disk drive after the next reboot.

If the specified threshold is exceeded, the Content Engine either records this event or reboots. If the automatic reload feature is enabled and this threshold is exceeded, then the ACNS system automatically reboots the Content Engine. For more information about specifying this threshold, see the "Specifying the Disk Error-Handling Threshold" section on page 2-154.

Note You can also manually mark a disk drive as bad or good by using the **disk drive mark** EXEC command. For more information, see the "Manually Marking and Unmarking Content Engine Disk Drives" section on page 2-149.

In the ACNS 5.2 software release, support for remapping bad (but unused) sectors on a SCSI drive was added. In the ACNS 5.3 software release, this capability is extended to include IDE and Serial Advanced Technology Attachment [SATA] drives. For more information, see the *Cisco ACNS Software Update and Maintenance Guide.*

Specifying the Disk Error-Handling Threshold

In the ACNS 5.2 software and later releases, you can configure a disk error-handling threshold. This threshold determines how many disk errors can be detected before the disk drive is automatically marked as bad. By default, this threshold is set to 10.

The **disk error-handling threshold** option determines how many disk errors can be detected before the disk drive is automatically marked as bad. By default, this threshold is set to 10.

To change the default threshold, use the **disk error-handling threshold** global configuration command. Specify 0 if you never want the disk drive to be marked as bad.

If the bad disk drive is a critical disk drive, and the automatic reload feature (**disk error-handling reload** command) is enabled, then the ACNS software marks the disk drive as bad and the Content Engine is automatically reloaded. After the Content Engine is reloaded, a syslog message and an SNMP trap are generated.

By default, the automatic reload feature is disabled on a Content Engine. To enable the automatic reload feature, use the **disk error-handling reload** global configuration command. After enabling the automatic reload feature, use the **no disk error-handling reload** global configuration command to disable it.

Examples

The following example shows that five disk drive errors for a particular disk drive (for example, disk00) will be allowed before the disk drive is automatically marked as bad:

```
ContentEngine(config)# disk error-handling threshold 5
```

Related Commands

disk (EXEC mode)
show disks
show disks details

distribution

To reschedule and refresh content redistribution through multicast for all channels, or for a specified channel ID or name, use the **distribution** EXEC command.

distribution {**failover** {**channel-id** *channel-num* | **channel-name** *name*} [**force**] | **fallback** {**channel-id** *channel-num* | **channel-name** *name*}}

distribution multicast {**resend** {**all** | **channel-id** *channel-num* [**object** *url* | **on-demand-only**] | **channel-name** *name* [**object** *url* | **on-demand-only**]} | **send-nack-now** | **stop** {**all** | **channel-id** *channel-num* [**object** *url* | **on-demand-only**] | **channel-name** *name* [**object** *url* | **on-demand-only**]}

distribution primary-ip-fallback {**forwarder-id** *forwarder-num* | **forwarder-name** *name*}

distribution refresh {**meta-data channel-id** *channel-num* | **object** *object-url*}

Syntax Description

failover	Triggers the root or forwarder Content Engine to fail over and make this Content Engine the temporary root Content Engine. **Note** In the ACNS software, Release 5.3, the **fallover** option has been removed and replaced by the **failover** option.
channel-id	Specifies the channel ID to be used.
channel-num	Channel number (0–4294967295).
channel-name	Specifies the channel name descriptor to be used.
name	Channel name.
force	(Optional) Forces a failover regardless of whether the root or forwarder Content Engine is active.
fallback	Forces the temporary root Content Engine to become a receiver Content Engine.
multicast	Resends or stops a multicast distribution.
resend	Resends the content to all channels or a specified channel ID or name.
all	Resends the multicast distribution to all channels.
channel-id	Specifies the channel ID to be used in the multicast redistribution.
object	(Optional) Specifies the URL of the object to be resent.
url	URL of the object to be sent.
on-demand-only	(Optional) Triggers a resend for the specified content only when a NACK is issued.
channel-name	Specifies the channel name descriptor to be used in the multicast redistribution.
name	Channel name.
send-nack-now	Generates a negative acknowledgement (NACK) for incomplete objects and sends it to the multicast sender immediately.
stop	Stops a multicast redistribution.
all	Stops multicast redistribution of content to all channels.
channel-id	Stops multicast redistribution of content based on channel ID.

channel-num	Channel number (0–4294967295).
channel-name	Stops the multicast redistribution of the content based on the channel name.
name	Channel name.
primary-ip-fallback	Triggers the downstream receiver Content Engines to contact a forwarder using the forwarder's primary IP address. For more information, see the "distribution primary-ip-fallback Command" section on page 2-159.
forwarder-id	Specifies the forwarder Content Engine ID that is contacted by the receiver Content Engine.
forwarder-num	Forwarder Content Engine ID.
forwarder-name	Specifies the name of the forwarder Content Engine that is contacted by the receiver Content Engine.
name	Forwarder Content Engine name.
refresh	Forces the redistribution of content to be refreshed on every Content Engine.
meta-data	Forces the redistribution of metadata to be refreshed on every Content Engine.
channel-id	Specifies the channel ID to be used in the multicast distribution.
channel-num	Channel number (0–4294967295).
object	Forces the distribution of objects to be refreshed on every Content Engine.
object-url	Specifies the object URL that needs to be refreshed on every Content Engine.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

When the root Content Engine fails, use the **distribution failover** EXEC command on a Content Engine that is going to be the temporary root Content Engine to trigger an immediate failover to the temporary root Content Engine if you do not want to wait for the automatic failover process to occur. When you enter this command, the current Content Engine becomes the temporary root Content Engine if its forwarder is an inactive root Content Engine. If the root Content Engine has not failed, a failover to the temporary root Content Engine does not occur if you use the **distribution failover** EXEC command. Use the **distribution failover force** command to force a failover even if the root Content Engine is active.

Note In the ACNS software, Release 5.3, the **fallover** option has been removed and replaced by the **failover** option.

Use the **distribution fallback** command on a Content Engine that is currently the temporary root Content Engine to cause it to become a receiver Content Engine.

The **distribution multicast resend** EXEC command can be used to reschedule content redistribution through multicast for all channels or for a specified channel ID or name. This command is especially useful in satellite environments where the preconfigured multicast might fail because of weather conditions. This command internally resets the number of carousel passes completed to zero for the

desired target. Because the first resend of content always happens automatically without being triggered by a NACK, setting the carousel pass count to zero triggers a resend of the content. However, if the **on-demand-only** option is additionally specified, the first carousel pass made after the command has been issued can only be triggered by a NACK for the content involved. The **distribution multicast resend** EXEC command options are as follows:

- **distribution multicast resend all**—Redistributes the content using multicast for all channels.
- **distribution multicast resend** [**channel-id** *channel-num*]—Redistributes the content using multicast for the specified channel ID.
- **distribution multicast resend** [**channel-name** *channel-name*]—Redistributes the content using multicast for the specified channel name.

The **distribution multicast stop** EXEC command can be used to stop the multicast distribution for all channels or for a specified channel ID or name. The command options are as follows:

- **distribution multicast stop all**—Stops multicast distribution for all channels.
- **distribution multicast stop** [**channel-id** *channel-num*]—Stops multicast distribution for the specified channel ID.
- **distribution multicast stop** [**channel-name** *channel-name*]—Stops multicast distribution for the specified channel name.

Once stopped, you can restart multicast distribution using the **distribution multicast resend** EXEC command.

Use the **distribution refresh meta-data** {**channel-id** *channel-num*} command to request that the metadata receiver repeat a previous request for all the content metadata for the specified channel from its forwarder Content Engine. This method allows you to start over if the metadata receiver fails to replicate some metadata properly. The content metadata (machine-readable information that describes the characteristics of the content) must be distributed to a receiver first before the content can be replicated. The content metadata helps to define what content to retrieve, how content is retrieved, how recently the content has been updated, how the content is to be pre-positioned (for example, expiration time), and so forth. The metadata is always distributed using unicast. The content, however, can be replicated using either multicast or unicast. A multicast receiver rejects the multicast sender's advertisement of a file if the proper content metadata has not arrived.

Use the **distribution refresh object** *object-url* command to reissue a request for unicast or multicast distribution of the specified object. This command lets you obtain a new copy of an object if there is a corrupted copy on the Content Engine. After you enter this command, if the distribution is unicast, the unicast receiver reissues the request to its forwarder Content Engine. If the distribution is multicast, the Content Engine sends a NACK regarding this object to the multicast sender. The old content on the Content Engine is removed and a new copy is replicated.

NACK Interval Multiplier

To identify missing content and trigger a resend of a file, receiver Content Engines send a negative acknowledgement (NACK) message to the sender Content Engine. NACK messages generated by many receiver Content Engines could generate more traffic than the sender can handle. The ACNS 5.1 software release and later releases allow you to adjust the average interval between NACKs by configuring a NACK interval multiplier for an individual receiver Content Engine. This value (an integer between 0.1–10) adjusts the default average NACK interval (the default is 20 minutes) by the value configured as the interval multiplier. For example, if you set the NACK interval multiplier to 3, the interval between NACKs becomes 20 minutes x 3, or 60 minutes. This adjustment can be made as needed by choosing **Devices > Devices > Prepositioning > Distribution** in the Content Distribution Manager GUI.

To send an immediate NACK request rather than wait for the scheduled interval, you can use the **distribution multicast send-nack-now** EXEC command on a multicast receiver Content Engine.

distribution primary-ip-fallback Command

When downstream receiver Content Engines at the edge of the network try to access a forwarder Content Engine that is inside a NAT firewall, those receiver Content Engines that are inside the same NAT use one IP address (called the inside local IP address) to access the forwarder, but other receiver Content Engines that are outside the NAT need to use a different forwarder's IP address (called the inside global IP address or NAT address) to access the forwarder. A forwarder Content Engine registers the IP address configured on its primary interface with the Content Distribution Manager, and the Content Distribution Manager uses the primary IP address for communication with devices in the ACNS network. If the registered primary IP address is the inside local IP address and the forwarder is behind a NAT firewall, a receiver that is not inside the same NAT as the forwarder cannot contact it without special configuration. All other receivers inside the NAT use the inside local IP address to contact the forwarder that resides inside the NAT.

In releases of the ACNS software prior to 5.3, the unicast content distribution between two Content Engines always uses the device's primary IP address. In many cases, the primary IP address may not be an external IP address (inside global IP address or NAT address) and other devices that are not behind the same NAT can use the primary IP address to initiate communication with the forwarder directly. If a forwarder is inside a NAT firewall and its primary IP address is not an external IP address, the unicast receiver and the metadata receiver cannot contact the forwarder. Therefore, content metadata replication and unicast content replication do not work. This method imposes a restriction that unicast content distribution cannot occur across NAT firewalls in releases of the ACNS software earlier than 5.3.

In releases of the ACNS software prior to 5.3, the subscribing receiver Content Engine that is not multicast-enabled uses unicasting to poll the metadata and content from the forwarder Content Engine using the forwarder's primary IP address. In the ACNS 5.3 software release, NAT (see the "NAT Firewall" section on page 2-159 for more information) support for unicast distribution was added. When the receiver Content Engine polls its forwarder from an upstream location for the content metadata or content, the receiver first connects to the forwarder using the forwarder's primary IP address. If it fails and the forwarder's NAT address has been configured, then the unicast receiver tries to poll the forwarder using the forwarder's NAT address. If the receiver polls the forwarder successfully using the NAT address, the receiver continues to use the forwarder's NAT address during the subsequent polling intervals with the same forwarder. The unicast receiver retries to connect to the forwarder using the forwarder's primary IP address only after one hour. Even if the unicast receiver is able to poll the forwarder using the forwarder's primary IP address, it would take one hour for the receiver to fall back to the forwarder's primary IP address automatically. You can use the **distribution primary-ip-fallback** command to enable the receiver that is using the NAT address of the forwarder to fall back to the primary IP address immediately, if you are certain that the forwarder's primary IP address is working.

NAT Firewall

Network Address Translation (NAT) enables private IP internetworks that use nonregistered IP addresses to connect to the Internet. NAT is configured on the firewall at the border of a stub domain (referred to as the inside network) and a public network such as the Internet (referred to as the outside network). NAT translates the internal local addresses to globally unique IP addresses before sending packets to the outside network. You can configure NAT to advertise only one address for the entire network to the outside world. This configuration provides additional security, effectively hiding the entire internal network from the world behind that address. NAT has the dual functionality of security and address conservation and is typically implemented in remote access environments.

In the inside network's domain, hosts have addresses in the one address space. While on the outside, they appear to have addresses in another address space when NAT is configured. The first address space is referred to as the local address space while the second is referred to as the global address space.

Hosts in outside networks can be subject to translation and can have local and global addresses.

NAT uses the following definitions:

- Inside local address—The IP address that is assigned to a host on the inside network. The address is probably not a legitimate IP address assigned by the Network Information Center (NIC) or service provider.
- Inside global address—A legitimate IP address (assigned by the NIC or service provider) that represents one or more inside local IP addresses to the outside world.
- Outside local address—The IP address of an outside host as it appears to the inside network. Not necessarily a legitimate address, it was allocated from an address space routable on the inside.
- Outside global address—The IP address assigned to a host on the outside network by the host's owner. The address was allocated from a globally routable address or network space.

Examples

The following example triggers resending of a particular file or the content to all channels:

```
ContentEngine# distribution multicast resend all
ContentEngine#
```

The following example triggers a NACK request immediately:

```
CONTENTENGINE# distribution multicast send-nack-now
Multicast NACK will be collected and sent immediatelly.
```

Related Commands

multicast
show statistics distribution

dns

To configure the Content Engine's DNS cache, use the **dns** global configuration command. To disable the DNS cache, use the **no** form of this command.

dns {**enable** | **listen** {*ip-address* **port** *port-num* **hostname** *hostname* | **all port** *port-num* **hostname** *hostname*} | **max-cache-memory** *max-mem* | **max-ttl** *ttl* | **min-ttl** *ttl* | **pin** {**both** *hostname ip-address* | **cname** *records* | **forward** *hostname ip-addresses* | **reverse** *hostname ip-address*} | **retry-period** *seconds* | **retry-timeout** *seconds* | **serial-lookup** | **use-expired enable** | **use-original-server** {**after-configured** | **before-configured** | **only**}

no dns {**enable** | **listen** *ip-address* **port** *port-num* **hostname** *hostname* | **all port** *port-num* **hostname** *hostname* | **max-cache-memory** *max-mem* | **max-ttl** *ttl* | **min-ttl** *ttl* | **pin** {**both** *hostname ip-address* | **cname** *records* | **forward** *hostname ip-addresses* | **reverse** *hostname ip-address*} | **retry-period** *seconds* | **retry-timeout** *seconds* | **serial-lookup** | **use-expired enable** | **use-original-server** {**after-configured** | **before-configured** | **only**}

Syntax Description

enable	Enables the Content Engine's DNS cache for resolution of DNS names to addresses.
listen	Configures the IP address and port number that the DNS cache uses to listen for requests.
ip-address	IP address on the host (the limit is 64).
port	Configures the DNS cache listener port number.
port-num	Port number (1–65535).
hostname	Configures the listener hostname to be mapped to the IP address.
hostname	Hostname of the listener.
all	Binds the DNS cache listener to any IP address on the host.
port	Configures the DNS cache listener port number.
max-cache-memory	Sets the maximum size of the cache memory.
max-mem	Maximum memory to be used in megabytes (5–512).
max-ttl	Specifies the maximum time to store a resource.
ttl	Time to Live in seconds (1–604800).
min-ttl	Specifies the minimum time to store a resource.
ttl	Time to Live in seconds (1–604800).
pin	Statically maps the IP addresses and hostnames. The maximum limit is 256.
both	Inserts bidirectional mapping.
hostname	Hostname of bidirectional mapping to IP address.
ip-address	IP address of bidirectional mapping.
cname	Inserts CNAME mapping.
records	Maps CNAME to the address (A) records. You can configure a maximum of eight records.
forward	Inserts forward mapping.
hostname	Hostname mapped to the forward IP address.
ip-addresses	Forward IP addresses. You can configure a maximum of eight IP addresses.

reverse	Inserts reverse mapping.
hostname	Hostname mapped to the reverse IP address.
ip-address	Reverse IP address.
retry-period	Sets the maximum time period before an unanswered request is discarded.
seconds	Maximum amount of time to wait before retries, in seconds (1–120).
retry-timeout	Sets the time in seconds between the request retries.
seconds	Time between requests in seconds (1–10).
serial-lookup	Queries each configured name server in turn if the previous request was unsuccessful.
use-expired	Uses a resource record in the DNS cache, even after it has expired.
enable	Enables the Content Engine to use a resource record in the DNS cache, even after it has expired.
use-original-server	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.
after-configured	Configures the DNS cache service on a Content Engine to try the configured DNS servers first and if they fail, then to try the original DNS server.
before-configured	Configures the DNS cache service on a Content Engine to try the original DNS server first, and then the configured DNS servers.
only	Configures the DNS cache service on a Content Engine to use only the list of configured DNS servers. This is the default.

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

DNS is used in the Internet for translating names of network nodes into IP addresses. DNS allows the network to translate domain names entered in requests into their associated IP addresses. For example, when end users (web clients) enter http://www.cisco.com into their browsers, DNS translates the domain name cisco.com into its associated IP address so that these requests can be processed (the requested content can be served to the web clients).

DNS caching allows the Content Engine to cache DNS entries to avoid multiple WAN accesses for DNS server resolution. When you enable DNS caching on a Content Engine, the Content Engine caches the results of recent DNS queries for faster resolution of identical queries in the future. This cached information is then available to clients making future requests. The ability to store DNS information that can then be distributed to requesting clients turns the Content Engine into a DNS caching name server.

Caution We recommend that you enable the DNS caching with WCCP interception on a Content Engine.

In centrally managed ACNS networks, configuring the DNS caching service with WCCP interception on a centrally managed Content Engine causes a conflict with the Content Router, because they both listen for DNS requests on the same port (port 53). They are mutually exclusive and you should not configure DNS cache support with WCCP interception in such environments. You can, however, enable the standard DNS caching service (without WCCP interception support) in centrally managed ACNS networks.

The structure and types of the DNS servers come in several different topologies as follows:

- All clients talk to either a single DNS server or a cluster of DNS servers as follows:
  - In a single DNS server, if the DNS query succeeds, then the resolved entry is returned; otherwise, the query is unsuccessful.
  - In a cluster of DNS servers, one DNS server functions as the primary DNS server. The request is queried against the primary DNS server. If the query succeeds, then the resolved entry is returned. If the query fails, and if DNS server clusters have been configured to use the recursive lookup method, these servers are queried in the order that they have been configured. If one query succeeds, the resolved entry is returned; otherwise, the query is unsuccessful.
- Clients talk to a local or regional DNS server (usually on the same LAN as the client), which forwards the request to a name server to resolve names. If recursive lookup has been configured on the name server, the configured name servers are queried repeatedly for name resolution.

Either the forwarding servers or the recursive server can use its cache to serve the request, depending on the availability of the resource record in the cache and its corresponding Time-To-Live (TTL) configuration.

To enable DNS caching on a Content Engine, you must complete the following tasks:

- Specify the list of DNS servers, which are used by the network to translate requested domain names into IP addresses that the Content Engine should use for domain name resolution.
- Specify the name of the local domain.
- Specify the DNS cache size; that is, the maximum number of records that the DNS cache on the Content Engine should store.
- Enable the WCCP Version 2 DNS caching service (the DNS service [service 53]) on the Content Engine.

The ACNS software, Release 5.1, supported the transparent interception of DNS requests using WCCP. To enable this feature, you must configure the WCCP Version 2 DNS caching service (service 53) on the Content Engine and the WCCP Version 2-enabled router. For more information, see the “Configuring DNS Servers for the DNS Caching Service (Service 53)” section.

Use the **dns enable** command to start the DNS server after the listener port is configured. Enabling the DNS server creates an entry of 127.0.0.1 as the name server for the system and starts the memory-based DNS cache. Use the **no** form of this command to disable the DNS cache.

The **dns listen** command configures the DNS server port to listen for new client queries and invokes query resolution routines. Once the hostname has been resolved to an IP address, it is stored in the memory-based DNS cache.

Note The domain name resolution requires that you configure at least one DNS name server on the Content Engine. You can configure one or more DNS name servers for the Content Engine by defining a list of DNS servers for the Content Engine through the **ip name-server** global configuration command.

It is important that you impose a strict maximum memory limit within which the DNS server operates so as not to unduly tax the overall system resources. Use the **dns max-cache-memory** command to set the maximum size of the cache memory.

The DNS server must know the DNS name of the host on which it is being enabled and map the name to an IP address within its own cache. If the **dns listen** command name does not match a DNS name, use the **pin** commands to pin an IP address to name mapping. The **dns pin** commands (**both**, **cname**, **forward**, and **reverse**) allow you to lock an IP address against a name within the cache. The **forward** command maps the hostname to the IP address. The **reverse** command maps the IP address to the hostname. The **both** command maps in both the forward and reverse directions. The **cname** command inserts CNAME mapping.

The **dns retry-period** command sets the time period before an unanswered request is discarded. The **dns retry-timeout** command sets the time between retransmission of UDP DNS requests sent to an upstream DNS server. Because the DNS protocol is using UDP packets that can be lost or dropped, the burden of retransmitting DNS requests is on the requester. Typically, a retransmit is initiated every 3 seconds until a response is received, or if a response is not received, the request times out after 60 seconds. If a DNS server times out, then a new upstream server is selected to query. If there are no more servers to query upstream, then the server returns a DNS failed response to the requesting client. The **dns serial-lookup command** causes each name server to be queried in turn if the previous request is unsuccessful.

Use the **dns use-original-server** command to control the response from the DNS caching name server when it receives transparent requests from a WCCP-enabled router. These transparent requests contain the original destination server in the packet headers. However, because the Content Engine CLI can have more than one DNS server configured, the **dns use-original-server** command allows you to use the original destination from the packet headers or the original destination obtained from the configured DNS name servers. You can also use this command to use a combination of destinations based on either the original server destination or one obtained from the configured DNS servers.

Examples

The following example shows how to configure the listener IP address, port number, and hostname. The DNS cache is then enabled.

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

The following example shows how to set the DNS cache retry timeout period:

```
ContentEngine(config)# dns retry-timeout 10
```

The following example shows how to set the size of the DNS cache to 20,000 records:

```
ContentEngine(config)# dns-cache size 20000
```

The following example shows how to set the length of time that must elapse before an unanswered request is discarded:

```
CONTENTENGINE(config)# dns retry-period 50
```

The following example shows how to set the interval between retransmission of UDP DNS requests that are sent to an upstream DNS server:

```
CONTENTENGINE(config)# dns retry-timeout 5
```

Related Commands

dnslookup
show dns
wccp dns

dnslookup

To resolve a host or domain name to an IP address, use the **dnslookup** EXEC command.

dnslookup {*hostname* | *domainname*}

Syntax Description

hostname	Name of host on the network.
domainname	Name of domain.

Defaults

No default behavior or values

Command Modes

EXEC

Examples

The following examples show that the **dnslookup** command is used to resolve the hostname *myhost* to IP address 172.31.69.11, *cisco.com* to IP address 192.168.219.25, and an IP address used as a hostname to 10.0.11.0:

```
ContentEngine# dnslookup myhost
official hostname: myhost.cisco.com
          address: 172.31.69.11

ContentEngine# dnslookup cisco.com
official hostname: cisco.com
         address: 192.168.219.25

ContentEngine# dnslookup 10.0.11.0
official hostname: 10.0.11.0
          address: 10.0.11.0
```

Related Commands

dns

enable

To access privileged EXEC commands, use the **enable** EXEC command.

enable

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

To access privileged EXEC mode from user EXEC mode, use the **enable** command. The **disable** command takes you from privileged EXEC mode to user EXEC mode.

Examples

The following example shows how to access privileged EXEC mode:

```
ContentEngine> enable
ContentEngine#
```

Related Commands

disable
exit

end

To exit global configuration mode, use the **end** global configuration command.

end

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

Use the **end** command to exit global configuration mode after completing any changes to the running configuration. To save new configurations to NVRAM, use the **write** command.

In addition, you can press **Ctrl-Z** to exit global configuration mode.

Examples

The following example shows how to exit global configuration mode:

```
ContentEngine(config)# end
ContentEngine#
```

Related Commands

exit

error-handling

To set error-handling options on the Content Engine, use the **error-handling** global configuration command. To undo the error handling options, use the **no** form of this command.

error-handling {**reset-connection** | **send-cache-error** | **transparent**}

no error-handling

Syntax Description

reset-connection	Resets the TCP connection without specifying any error.
send-cache-error	Sends a cache error.
transparent	Makes the Content Engine transparent to the client.

Defaults

The default is the **error-handling transparent** option.

Command Modes

global configuration

Usage Guidelines

The **error-handling transparent** option is set by default, so that the Content Engine will not send errors to the client but will bypass the client connections to the server. Setting the **error-handling send-cache-error** command will send a Content Engine-generated error page to the client. Using the **reset-connection** option resets the TCP client connection.

Note Setting error handling options on a Content Engine applies only to HTTP proxy caching.

If error handling is set to **transparent**, the Content Engine adds the client/server pair to the WCCP bypass list. The Content Engine will send a retry message to the client. The retried connection from the client is then bypassed by the Content Engine.

A transparent error bypass is triggered only if the following conditions exist:

- The Content Engine is configured to preserve transparency as opposed to preserving confinement and control.
- The transaction is transparently intercepted.
- WCCP (WCCP Version 2 or later) on the Content Engine is capable of performing a bypass.

For a client request, the bypass occurs under the following conditions:

- If the request is malformed and fails to parse
- If the client is denied access
- If the client fails proxy authentication

For a server response, the bypass occurs under the following conditions:

- If the response is not obtained explicitly through an outgoing proxy
- If the request is malformed and fails to parse
- If the request has a 501, 502, 503, 504, or 505 status code, which may indicate that an error exists on the server

With the **transparent** option enabled, end users can receive browser-generated messages rather than a Content Engine-generated HTML page for errors that the Content Engine encounters while processing a client request or response. The Content Engine remains transparent (invisible) to the end user.

Transparent error reporting is implemented as follows:

- Content Engine running WCCP Version 2

  To make the source of the error messages transparent to the user, the client/server pair is added to the bypass list and an HTTP redirect message is sent to the client, requesting the client to redirect the request to the same URL as before. The client, on receiving the redirect message, sends back the request once again. This time, the request is bypassed by the Content Engine because the client/server pair is on the bypass list. The request now goes to the server directly. Because the connection was not accepted by the Content Engine, any timeout error, failure to connect to the server, or mangled response from the server is handled by the browser. Currently, all entries on the bypass list are kept for a configurable period of time (the default is 20 minutes).

  With the **reset-connection** option, a reset is sent back to the client and the connection is closed if it encounters an error from the server. When a browser receives a connection reset, it displays a Connection Reset By Peer alert box.

- Content Engine running WCCP Version 1

  For all error conditions, the Content Engine sends back a reset and closes the connection. It does not send back any error pages. All errors seen by the clients are in the familiar browser error format.

- Content Engine acting as an incoming proxy server

  The Content Engine sends back HTML error pages. When clients are using the Content Engine as an incoming proxy server, they receive the HTML error pages generated by the Content Engine.

Examples

The following example shows how to configure transparent error handling on the Content Engine:

```
ContentEngine(config)# error-handling transparent
```

Related Commands

http proxy incoming

exception

To enable error handling or debug mode, use the **exception debug** global configuration command. To revert to the default value, use the **no** form of this command.

exception {**coredump** | **debug**}

no exception {**coredump** | **debug**}

Syntax Description

coredump	Causes the proxy processes to do a core dump if the process crashes.
debug	Causes the proxy processes to hang if the process crashes, until they are explicitly killed.

Defaults

The default is disabled.

Command Modes

global configuration

Usage Guidelines

Note We recommend that you use the **exception debug** and **exception coredump** commands only at the direction of Cisco TAC (see the "Obtaining Technical Assistance" section on page 17). Content Engine performance is affected when you run the **exception debug** or **exception coredump** command.

Examples

The following example causes proxy processes to hang if the system crashes:

```
ContentEngine(config)# exception debug
```

The following example disables debug mode for exceptions:

```
ContentEngine(config)# no exception debug
```

Related Commands

debug

exec-timeout

To configure the length of time that an inactive Telnet or Secure Shell (SSH) session remains open, use the **exec-timeout** global configuration command. To revert to the default value, use the **no** form of this command.

exec-timeout *timeout*

no exec-timeout

Syntax Description

timeout	Timeout in minutes (0–44640).

Defaults

The default is 15 minutes.

Command Modes

global configuration

Usage Guidelines

A Telnet or SSH session with the Content Engine can remain open and inactive for the interval of time specified by the **exec-timeout** command. When the **exec-timeout** interval elapses, the Content Engine automatically closes the Telnet or SSH session.

Configuring a timeout interval of 0 minutes by entering the **exec-timeout 0** command is equivalent to disabling the session-timeout feature.

Examples

The following example configures a timeout of 100 minutes:

```
ContentEngine(config)# exec-timeout 100
```

The following example negates the configured timeout of 100 minutes and reverts to the default value of 15 minutes:

```
ContentEngine(config)# no exec-timeout
```

Related Commands

telnet enable
sshd

exit

To access the EXEC command shell from the global, interface, and debug configuration command shells, use the **exit** command.

exit

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC, global configuration, and interface configuration

Usage Guidelines

Use the **exit** command in any configuration mode to return to EXEC mode. Using this command is equivalent to pressing the **Ctrl-Z** key or entering the **end** command.

The **exit** command issued in the user-level EXEC shell terminates the console or Telnet session. You can also use the **exit** command to exit other configuration modes that are available from the global configuration mode for managing specific features (see the commands marked with a footnote in Table 2-1).

Examples

The following example terminates global configuration mode and returns to the privileged-level EXEC mode:

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

The following example terminates privileged-level EXEC mode and returns to the user-level EXEC mode:

```
ContentEngine# exit
ContentEngine>
```

Related Commands

end

external-ip

To configure up to eight external Network Address Translation (NAT) IP addresses, use the **external-ip** global configuration command. To remove the NAT IP addresses, use the **no** form of this command.

external-ip *ip-addresses*

no external-ip *ip-addresses*

Syntax Description

ip-addresses	A maximum of eight external or NAT IP addresses can be configured.

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

Use this command to configure up to eight Network Address Translation IP addresses to allow the router to translate up to eight internal addresses to registered unique addresses and translate external registered addresses to addresses that are unique to the private network. If the IP address of the RTSP gateway has not been configured on the Content Engine, then the external IP address is configured as the IP address of the RTSP gateway.

In an ACNS network, there are two methods for a device registered with the Content Distribution Manager (Content Engines, Content Routers, or the standby Content Distribution Manager) to obtain configuration information from the primary Content Distribution Manager. The primary method is for the device to periodically poll the primary Content Distribution Manager on port 443 to request a configuration update. You cannot configure this port number. The backup method is when the Content Distribution Manager pushes configuration updates to a registered device as soon as possible by issuing a notification to the registered device on port 443. This method allows changes to take effect in a timelier manner. You cannot configure this port number even when the backup method is being used. ACNS networks do not work reliably if devices registered with the Content Distribution Manager are unable to poll the Content Distribution Manager for configuration updates. When a receiver Content Engine requests the content and content metadata from a forwarder Content Engine, it contacts the forwarder Content Engine on port 443.

When a device (Content Engines at the edge of the network, Content Routers, and primary or standby Content Distribution Managers) is inside a NAT firewall, those devices that are inside the same NAT use one IP address (the inside local IP address) to access the device and those devices that are outside the NAT use a different IP address (the NAT IP address or inside global IP address) to access the device. A centrally managed device advertises only its inside local IP address to the Content Distribution Manager. All other devices inside the NAT use the inside local IP address to contact the centrally managed device that resides inside the NAT. A device that is not inside the same NAT as the centrally managed device cannot contact it without a special configuration.

If the primary Content Distribution Manager is inside a NAT, you can allow a device outside the NAT to poll it for getUpdate requests by configuring a static translation (NAT IP address or inside global IP address) for the Content Distribution Manager's inside local IP address on its NAT, and using this address, rather than the Content Distribution Manager's inside local IP address in the **cdm ip** *ip-address* global configuration command when you register the device to the Content Distribution Manager. If a

Content Engine or Content Router is inside a NAT and the CDM is outside the NAT, you can allow the Content Engine or Content Router to poll for getUpdate requests by configuring a static translation (NAT IP address or inside global IP address) for the Content Engine or Content Router's inside local address on its NAT.

Note Static translation establishes a one-to-one mapping between your inside local address and an inside global address. Static translation is useful when a host on the inside must be accessible by a fixed address from the outside.

Examples

The following example configures four external NAT IP addresses:

```
ContentEngine(config)# external-ip 192.168.43.1 192.168.43.2 192.168.43.3 192.168.43.4
```

Related Commands

interface
ip

find-pattern

To search for a particular pattern in a file, use the **find-pattern** EXEC command.

find-pattern {**binary** *reg-express filename* | **case** {**binary** *reg-express filename* | **count** *reg-express filename* | **lineno** *reg-express filename* | **match** *reg-express filename* | **nomatch** *reg-express filename* | **recursive** *reg-express filename*} | **count** *reg-express filename* | **lineno** *reg-express filename* | **match** *reg-express filename* | **nomatch** *reg-express filename* | **recursive** *reg-express filename*}

Syntax Description

binary	Does not suppress the binary output.
reg-express	Regular expression to be matched.
filename	Filename.
case	Matches the case-sensitive pattern.
count	Prints the number of matching lines.
lineno	Prints the line number with output.
match	Prints the matching lines.
nomatch	Prints the nonmatching lines.
recursive	Searches a directory recursively.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to search for a particular regular expression pattern in a file.

Examples

The following example searches a file recursively for a case-sensitive pattern:

```
ContentEngine# find-pattern case recursive admin removed_core
-rw-------    1 admin    root     95600640 Oct 12 10:27 /local/local1/core_dir/c
ore.5.2.1.b5.eh.2796
-rw-------    1 admin    root     97054720 Jan 11 11:31 /local/local1/core_dir/c
ore.cache.5.3.0.b131.cnbuild.14086
-rw-------    1 admin    root     96845824 Jan 11 11:32 /local/local1/core_dir/c
ore.cache.5.3.0.b131.cnbuild.14823
-rw-------    1 admin    root     101580800 Jan 11 12:01 /local/local1/core_dir/
core.cache.5.3.0.b131.cnbuild.15134
-rw-------    1 admin    root     96759808 Jan 11 12:59 /local/local1/core_dir/c
ore.cache.5.3.0.b131.cnbuild.20016
-rw-------    1 admin    root     97124352 Jan 11 13:26 /local/local1/core_dir/c
ore.cache.5.3.0.b131.cnbuild.30249
-rw-------    1 admin    root     98328576 Jan 11 11:27 /local/local1/core_dir/c
ore.cache.5.3.0.b131.cnbuild.8095
```

The following example searches a file for a pattern and prints the matching lines:

```
ContentEngine# find-pattern match 10 removed_core
Tue Oct 12 10:30:03 UTC 2004
-rw-------    1 admin    root     95600640 Oct 12 10:27 /local/local1/core_dir/c
ore.5.2.1.b5.eh.2796
-rw-------    1 admin    root     101580800 Jan 11 12:01 /local/local1/core_dir/
core.cache.5.3.0.b131.cnbuild.15134
```

The following example searches a file for a pattern and prints the number of matching lines:

```
ContentEngine# find-pattern count 10 removed_core
3
```

Related Commands

cd
dir
ls
lls

ftp-native

To configure FTP native caching services on the Content Engine, use the **ftp-native** global configuration command. Use the **no** form of this command to selectively disable options.

ftp-native custom-message download {**welcome** *welcome-message url* | **acl-denied** *acl-denied-message url*} | **reset** {**acl-denied** | **welcome** | **all** } | **upload** {*ip-address* | *hostname*} *dirname filename message*

ftp-native object max-size *size*

ftp-native proxy {**active-mode enable** | **incoming** *ports*}

no ftp-native {**object max-size** | **proxy** {**active-mode enable** | **incoming** [*ports*]} | **custom-message download**}

Syntax Description

download	Copies the custom message file to the Content Engine from the specified URL. To change the text for a specific message, use this option to identify the message that you want to change, and specify the URL that is the source for the custom message file. The custom message file can be up to 16 KB in size and is used instead of the standard message for the specified message.
welcome	Indicates that you want to download the custom welcome message for the FTP proxy-mode welcome message.
welcome-message url	URL from which the custom message file (the file that contains the FTP proxy-mode welcome message or the ACL access denied error message) should be retrieved. The file size cannot exceed 16 KB.
acl-denied	Indicates that you want to download the custom error message that the Content Engine is to display to the FTP client because the client is being denied access based on the IP ACLs that have been defined for the FTP application.
reset	Specifies that the Content Engine (the FTP proxy) is to revert to the default message. Also deletes the local message files on the Content Engine.
acl-denied	Specifies that the Content Engine is to revert to the default ACL access-denied error message.
welcome	Specifies that the Content Engine is to revert to the default FTP proxy-mode welcome message.
all	Specifies that the Content Engine is to revert to all default FTP proxy messages.
upload	Uploads the custom message file from the Content Engine to the specified host, directory, and file.
ip-address	IP address of the host to which the Content Engine is to copy the custom message file.
hostname	Hostname to which the Content Engine is to copy the file that contains the custom message. The host should be reachable and allow copying a file to the specified directory.
dirname	Directory name to which to copy the custom message file.
filename	Filename to which to copy the custom message file.

object	Sets the configuration of FTP objects for FTP native caching.
max-size	Sets the maximum size of a cacheable object for FTP native caching.
size	Maximum size of an FTP object in kilobytes (KB) that should be stored in the Content Engine cache for FTP native caching (1–2096128).
proxy	Sets the proxy configuration parameters for the FTP native requests.
active-mode	Configures FTP native active mode to fetch files.
enable	Enables FTP native active mode on the Content Engine.
incoming	Sets the incoming port for the proxy-mode FTP native requests.
ports	Ports to listen for requests (1–65535). You can configure a maximum of eight ports.

Defaults

Active mode is disabled by default.

object max-size: 2096128 KB

Command Modes

global configuration

Usage Guidelines

A Content Engine that operates as an FTP proxy supports passive and active mode for fetching files and directories. The FTP-native service (service 60) is the WCCP caching service that permits WCCP Version 2 routers to redirect transparent FTP native requests transparently to a single port on the Content Engine. The Content Engine retrieves the requested FTP content, stores a copy locally (performs FTP native caching), and serves the requested content to the client.

Note The settings configured using the **ftp-native** global configuration command apply to both transparent-mode and proxy-mode FTP native caching. See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure transparent FTP native caching.

In FTP native caching mode, if the **ftp-native proxy active-mode enable** global configuration command is specified, then the Content Engine uses the same mode with the FTP server for the data connection as the client used to access the Content Engine, which can be either active or passive:

```
ContentEngine(config)# ftp-native proxy active-mode ?
  enable  Adhere to client's mode for native FTP
```

Note For more information about nontransparent proxy for the FTP native mode feature, see the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

In the ACNS 5.3 software release, support for the proxying and caching of nontransparent FTP native requests was added. With this new capability, a Content Engine can handle client FTP native requests from FTP clients in proxy mode.

When the Content Engine receives an FTP native request from an FTP client (for example, an FTP native request from a Reflection X or WS-FTP client or a UNIX or DOS command-line FTP program), the Content Engine processes the request. If the requested content is already stored in the local cache, the Content Engine serves the content to the FTP client. Otherwise, the Content Engine performs an FTP

request to the origin FTP server to retrieve the requested content and then stores the content in its local cache. This type of caching is called nontransparent FTP native caching. Native FTP requests are logged in the HTTP transaction log on the Content Engine.

Both passive and active mode for retrieving files and directories are supported. In FTP native caching mode, if you use the **ftp-native proxy active-mode enable** global configuration command, then the Content Engine uses the same mode with the FTP server for the data connection that the client used to access the Content Engine, which can be either active or passive. If you do not specify the **ftp-native proxy active-mode enable** command, the Content Engine uses passive mode with the FTP server for the data connection.

Note A passive mode transfer between the FTP client and the FTP native proxy cannot occur if the Content Engine is behind a NAT because the Content Engine does not know its conduit IP address.

Use the **ftp-native proxy incoming** *ports* global configuration command to configure the port numbers for the incoming proxy-mode FTP requests (FTP native requests).

Note If you use the **no ftp-native proxy incoming** *ports* command, all the proxy ports configured for FTP native caching are disabled.

Use the **ftp-native object max-size** global configuration command to specify the maximum size of an FTP object that should be stored in the Content Engine cache for FTP native caching. You can configure this parameter for particular objects in the cache. However, this parameter does not apply to directory listings. The FTP native proxy does not cache directory listings; it only services the request and response between the client and the server. See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for a description on how to configure nontransparent FTP native caching on Content Engines.

Configure the FTP clients (client side) to send their FTP native requests directly to the Content Engine. For more information, see the next section, "Configuring the Client Side of Nontransparent FTP Native Caching."

On the Content Engine, enter the **show ftp-native** EXEC command to view the current FTP native proxy configuration. On the Content Engine, enter the **show statistics ftp-native** EXEC command to display statistics for the FTP native requests that this Content Engine has handled. To clear FTP native statistics on the Content Engine, enter the **clear statistics ftp-native** EXEC command.

Note In the ACNS 5.3 software release, the **show ftp proxy** EXEC command was replaced with the **show ftp-native** and **show ftp-over-http** EXEC commands.

In the ACNS 5.3 software release, the **show statistics ftp** EXEC command was replaced with the **show statistics ftp-over-http** and **show statistics ftp-native** EXEC commands. In the ACNS 5.3 software release, the **clear statistics ftp** EXEC command was replaced with the **clear statistics ftp-over-http** and **clear statistics ftp-native** EXEC commands.

Configuring the Client Side of Nontransparent FTP Native Caching

Content Engines that act as nontransparent FTP proxy servers can accept FTP native requests from such FTP clients as Reflection X clients, WS-FTP clients, and UNIX or DOS command-line FTP programs. See the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments* for information on how to configure the client-side proxy FTP request to the Content Engine.

Examples

The following example sets the maximum size for an FTP object size to 2 MB for FTP native caching:

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

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

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

The following example disables all the proxy ports for FTP native caching:

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

The following two examples show the use of native FTP with a Content Engine. In the first example, the user logs in with an actual username name (huff) and is able to retrieve the requested file (test.c) from the FTP server. In this case, the home directory for the user named huff is /home/huff.

```
ContentEngine# ftp server.cisco.com
Connected to server.cisco.com.
220 server.cisco.com FTP server (Version wu-2.6.0(1) Mon Feb 28 10:30:36 EST 2000) ready.
Name (server:huff): huff
331 Password required for myserver.
Password:
230 User huff logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/home/huff" is current directory.
ftp> get /tmp/test.c
200 PORT command successful.
150 Opening BINARY mode data connection for /tmp/test.c (645 bytes).
226 Transfer complete.
645 bytes received in 0.00077 seconds (8.2e+02 Kbytes/s)
ftp> quit
ContentEngine#
```

The following example shows the user logging in as an anonymous user and not being able to retrieve the requested file (test.c) because the file is not located in the document root directory of the FTP server (/), which is the home directory for any anonymous user:

```
ContentEngine# ftp server.cisco.com
Connected to server.cisco.com.
220 server.cisco.com FTP server (Version wu-2.6.0(1) Mon Feb 28 10:30:36 EST 2000) ready.
Name (server:huff): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: test@cisco.com
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/" is current directory.
ftp>
ftp> passive
Passive mode on
ftp> get
(remote-file) /tmp/test.c
(local-file) test.c
local: test.c remote: /tmp/test.c
227 Entering Passive Mode (172.31.255.255)
550 /tmp/test.c: No such file or directory.
ftp>
ContentEngine#
```

The following example displays a list of the names of the configured FTP native custom messages:

```
ContentEngine# show ftp-native custom-message
```

The following example displays the contents of the local copy of the specified custom message (for example, the acl-denied message or the welcome message that has been downloaded to the Content Engine) on the CLI display screen:

```
ContentEngine# show ftp-native custom-message
```

The following example copies the FTP native custom welcome message to the Content Engine:

```
ContentEngine# ftp-native custom-message download welcome
http://www.myserver.com/errors/ftp-native-welcome.txt
```

The following examples show how to configure the port numbers for the incoming proxy-mode FTP requests (FTP native requests).

```
CONTENTENGINE# ftp-native proxy incoming ports
```

where the *ports* are the port numbers on which the Content Engine accepts incoming requests (native FTP requests) from FTP clients. Valid port numbers are 1–65535. You can specify up to eight incoming ports.

The following example shows how to configure the Content Engine to accept FTP native requests from FTP clients on 8 ports (port 8501, 8502, 8503, 8504, 8505, 8506, 8507, and 8508). You can configure up to eight incoming proxy ports on the same command line, as shown in the following example:

```
ContentEngine(config)# ftp-native proxy incoming 8501 8502 8503 8504 8505 8506 8507 8508
```

If you reenter the **ftp-native proxy incoming** command, the Content Engine only uses the ports specified in the most recently specified **ftp-native proxy incoming** command. For example, if you enter the **ftp-native proxy incoming** command, the Content Engine uses the eight specified ports as incoming proxy ports as follows:

```
ContentEngine(config)# ftp-native proxy incoming 8501 8502 8503 8504 8505 8506 8507 8508
```

However, if you reenter the **ftp-native proxy incoming** command, the Content Engine uses only port 8501 as an incoming proxy port and drops the other 7 previously configured ports as incoming proxy ports as follows:

```
ContentEngine(config)# ftp-native proxy incoming 8501
```

If you enter an illegal port number, an error message displays the information. The following example shows that if you specify port 554 as the incoming port for proxy-mode FTP native requests, you are informed that this port is reserved for the RTSP gateway that runs on the Content Engine:

```
ContentEngine(config)# ftp-native proxy incoming 554
Port 554 is reserved for application the RTSP_Gateway.
```

The following example shows how to use a UNIX command-line FTP program to configure the client-side proxy FTP request to the Content Engine that is acting as the nontransparent FTP proxy server:

```
shell# ftp -d 2.9.192.11 8501
Connected to 2.9.192.11.
220-Welcome to FTP-proxy.
220-Login to origin server using the 'USER username@server-hostname' command,
or 220 Login to origin server using the 'SITE server-hostname' and
'USER username' commands.
Name (2.9.192.11:admin): smartuser@abchost.company.com
331 Password required for smartuser.
Password:
```

```
230 User smartuser logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
---> PORT 2,9,192,10,81,212
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 8
-rw-r--r-- 1 nobody abc 5496 Jan 31 2002 boot
-rw-r--r-- 1 nobody abc 143 Oct 24 2001 cop
226 Transfer complete.
ftp> quit
---> QUIT
shell#
```

Related Commands

clear statistics ftp-native
debug ftp-native
ftp-over-http
show ftp-over-http
show statistics ftp-native
wccp ftp-native

ftp-over-http

To configure FTP-over-HTTP caching services on the Content Engine, use the **ftp-over-http** global configuration command. To selectively disable options, use the **no** form of this command.

ftp-over-http age-multiplier directory-listing *dl-time* **file** *fo-time*

ftp-over-http max-ttl {**days directory-listing** *dlmax-days* **file** *fmax-days* | **hours directory-listing** *dlmax-hours* **file** *fmax-hours* | **minutes directory-listing** *dlmax-min* **file** *fmax-min* | **seconds directory-listing** *dlmax-sec* **file** *fmax-sec*}

ftp-over-http min-ttl *min-minutes*

ftp-over-http object max-size *size*

ftp-over-http proxy {**active-mode enable** | **anonymous-pswd** *passwd* | **incoming** *ports* | **outgoing** {**connection-timeout** *timeout* | **host** {*hostname* | *ip-address*} *port* [**primary**] | **monitor** *interval* | **origin-server**}}

ftp-over-http reval-each-request {**all** | **directory-listing** | **none**}

no ftp-over-http {**age-multiplier** | **max-ttl** | **min-ttl** | **object max-size** | **proxy** {**active-mode enable** | **anonymous-pswd** | **incoming** | **outgoing** {**connection-timeout** | **host** {*hostname* | *ip-address*} *port* [**primary**] | **monitor** | **origin-server**}} | **reval-each-request**}

Syntax Description

age-multiplier	Specifies the FTP-over-HTTP caching heuristic modifiers.
directory-listing	Specifies the heuristic modifier of directory listing objects for FTP-over-HTTP caching. **Note** The directory listing is a list of files and subdirectories in a directory for FTP-over-HTTP caching. An example of a directory listing is a list of all files and folders in the ftp://vista2 directory.
dl-time	Expiration time of directory listing objects as a percentage of their age (0–100). The default is 30.
file	Specifies the heuristic modifier of file objects for FTP-over-HTTP caching.
fo-time	Expiration time of file objects as a percentage of their age (0–100). The default is 60.
max-ttl	Sets the maximum Time To Live for objects in the cache for FTP-over-HTTP caching.
days	Sets the maximum Time-To-Live units in days.
directory-listing	Sets the maximum Time To Live for directory listing objects in days for FTP-over-HTTP caching.
dlmax-days	Maximum Time To Live in days for directory listing objects (1–1825). The default is 7 days.
file	Sets the maximum Time To Live for file objects in days for FTP-over-HTTP caching.
fmax-days	Maximum Time To Live in days (1–1825). The default is 3 days.
hours	Sets the maximum Time-To-Live units in hours.

directory-listing	Sets the maximum Time To Live for directory listing objects in hours for FTP-over-HTTP caching.
dlmax-hours	Maximum Time To Live for directory listing objects in hours (1–43800). The default is 72 hours.
file	Sets the maximum Time To Live for file objects in hours for FTP-over-HTTP caching.
fmax-hours	Maximum Time To Live for file objects in hours (1–43800). The default is 168 hours.
minutes	Sets the maximum Time-To-Live units in minutes.
directory-listing	Sets the maximum Time To Live for directory listing objects in minutes for FTP-over-HTTP caching.
dlmax-min	Maximum Time To Live for directory listing objects in minutes (1–2628000). The default is 4320 minutes.
file	Sets the maximum Time To Live for file objects in minutes for FTP-over-HTTP caching.
fmax-min	Maximum Time To Live for file objects in minutes (1–2628000). The default is 10080 minutes.
seconds	Sets the maximum Time-To-Live units in seconds for FTP-over-HTTP caching.
directory-listing	Sets the maximum Time To Live for directory listing objects in seconds for FTP-over-HTTP caching.
dlmax-sec	Maximum Time To Live for directory listing objects in seconds (1–157680000). The default is 259200 seconds.
file	Sets the maximum Time To Live for file objects in seconds for FTP-over-HTTP caching.
fmax-sec	Maximum Time To Live for file objects in seconds (1–157680000). The default is 604800 seconds.
min-ttl	Sets the minimum Time To Live for FTP objects in the cache for FTP-over-HTTP caching.
min-minutes	Minimum Time To Live in minutes for FTP objects in the cache (0–86400).
object	Sets the configuration of FTP objects for FTP-over-HTTP caching.
max-size	Sets the maximum size of a cacheable object for FTP-over-HTTP caching.
size	Maximum size of an FTP object in kilobytes (KB) that should be stored in the Content Engine cache for FTP-over-HTTP caching (1–2096128).
proxy	Sets the proxy configuration parameters for FTP-over-HTTP requests.
active-mode	Configures FTP-over-HTTP active mode to fetch files.
enable	Enables FTP-over-HTTP active mode on the Content Engine.
anonymous-pswd	Sets the anonymous password string for FTP-over-HTTP requests (for example, wwwuser@cisco.com).
passwd	Anonymous password. The default is anonymous@hostname.
incoming	Sets the incoming port for proxy-mode FTP-over-HTTP requests.
ports	Ports to listen for requests (1–65535). You can configure a maximum of eight ports.
outgoing	Sets the parameters to direct outgoing FTP-over-HTTP requests to another proxy server.

connection-timeout	Specifies the timeout value (in microseconds) used for probing outgoing proxy servers.
timeout	Timeout value (in microseconds) used for probing outgoing proxy servers (200–5000000).
host	Sets the outgoing FTP-over-HTTP proxy host parameters.
hostname	Hostname of the outgoing FTP-over-HTTP proxy.
ip-address	IP address of the outgoing FTP-over-HTTP proxy.
port	Port of the outgoing FTP-over-HTTP proxy (1–65535).
primary	(Optional) Makes the configured proxy the primary FTP-over-HTTP proxy server.
monitor	Specifies the interval at which the outgoing proxy servers are to be monitored.
interval	Monitoring interval in seconds (10–300).
origin-server	Specifies that the origin server must be used if all outgoing proxy servers fail.
reval-each-request	Sets the scope of revalidation for every FTP-over-HTTP request.
all	Revalidates all objects on every FTP-over-HTTP request.
directory-listing	Revalidates the directory listing objects on every FTP-over-HTTP request.
none	Does not revalidate for each FTP-over-HTTP request.

Defaults

Active mode is disabled by default.

dl-time: 30 percent

fo-time: 60 percent

dlmax-days: 7 days

fmax-days: 3 days

dlmax-hours: 72 hours

fmax-hours: 168 hours

dlmax-min: 4320 minutes

fmax-min: 10080 minutes

dlmax-sec: 259200 seconds

fmax-sec: 604800 seconds

min-minutes: 86400 minutes

Maximum size of cacheable object sent using HTTP: 2 GB

Maximum size of cacheable object sent using FTP: 2 GB

Command Modes

global configuration

Usage Guidelines

A Content Engine can be configured for FTP caching in the following two usage modes:

- FTP-over-HTTP mode—The Content Engine (acting as a nontransparent forward proxy server) caches the contents of the specified FTP URLs that are sent to it directly by clients who are using the HTTP protocol. This mode allows users to use their browsers running the HTTP protocol to send and receive files on remote FTP servers. For more information, see the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.
- Native FTP mode—The Content Engine caches the contents of the FTP request that are sent from clients in the native FTP protocol. In the ACNS 5.3 software and later releases, native FTP caching is supported in transparent and nontransparent proxy mode. (Native FTP caching was supported only in transparent proxy mode in the ACNS 5.1 and 5.2 software releases.) For more information, see the "ftp-native" section on page 2-178.

In both of these usage modes, the Content Engine uses FTP to retrieve and locally cache the content of the FTP requests. These two usage modes differ in the protocol used by the client to issue the FTP request. In FTP-over-HTTP mode, clients use their browsers (the HTTP protocol) to issue FTP requests. In native FTP mode, clients use native FTP to issue FTP requests, as shown in the following example:

```
ContentEngine# ftp server.cisco.com
```

Note For information on the usage modes and types of supported FTP caching, see Chapter 7 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

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

In the ACNS 5.3 software release, the **ftp** keyword was replaced with the **ftp-over-http** and **ftp-native** keywords to clearly differentiate between FTP native caching and FTP-over-HTTP caching.

FTP-over-HTTP Caching Support

In the ACNS 5.0 software release, support for the proxying and caching of FTP-style requests over HTTP in proxy mode was added. When the Content Engine is configured in proxy mode, it can handle FTP-style requests over HTTP transport. When the Content Engine receives an FTP request from a client, it processes the request by searching its cache. If the object is not in its cache, it retrieves the object from an upstream FTP proxy server if this proxy server has been configured, or it retrieves the object directly from the origin FTP server.

With nontransparent FTP-over-HTTP caching, the Content Engine is functioning as a nontransparent forward proxy server for FTP-over-HTTP requests from client browsers. The ACNS 5.1 software and later releases support proxying and caching of FTP URL client requests using proxy-mode HTTP requests when URLs specify the FTP protocol (for example, ftp://ftp.mycompany.com/ftpdir/ftp_file).

The following example of an FTP-over-HTTP request shows how the end user can use a browser to access public files from an FTP server:

```
ftp://ftp.funet.fi/pub/cbm/crossplatform/converters/unix/
```

For these requests, the client uses HTTP as the transport protocol with the Content Engine, and the Content Engine uses FTP with the FTP server. When the Content Engine receives an FTP request from the web client, it first looks in its cache. If the object is not in its cache, it fetches the object from an upstream FTP proxy server (if one is configured) or directly from the origin FTP server.

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

The FTP proxy supports commonly used MIME types, attaches the corresponding header to the client, chooses the appropriate transfer type (binary or ASCII), and enables the browser to open the FTP file with the configured application. For unknown file types, the proxy uses binary transfer as the default and instructs the browser to save the download file instead of opening it. The FTP proxy returns a formatted directory listing to the client if the FTP server replies with a known format directory listing. The formatted directory listing has full information about the file or directory and provides the ability for users to choose the download transfer type.

Configure the port numbers for the incoming proxy-mode FTP-over-HTTP requests by entering the **ftp-over-http proxy incoming** *ports* global configuration command.

If you use the **ftp-over-http proxy incoming** command to configure the Content Engine to accept FTP-over-HTTP requests on a port other than port 80, you must also configure the client browsers to send their FTP-over-HTTP requests to that port.

You can configure FTP cache object freshness settings for FTP-over-HTTP caching. These parameters can be configured for either directory listings or particular objects in the cache.

Tip With the ACNS 5.x software, you can balance the HTTP and FTP object freshness with the cache hit rate. The ACNS software default parameters are weighted in favor of securing fresh content over maximizing the cache hit rate (to avoid increasing the cache hit rate by serving stale content). Text objects refer to HTML pages. Binary objects refer to all other web objects, such as GIFs and JPEGs.

- Specify the maximum size of an FTP object that should be stored in the Content Engine cache for FTP-over-HTTP caching by entering the **ftp-over-http object max-size** global configuration command.
- Configure FTP-over-HTTP caching by entering the **ftp-over-http age-multiplier**, **ftp-over-http max-ttl**, **ftp-over-http reval-each-request**, and the **ftp-over-http min-ttl** global configuration commands.
- Force the Content Engine to revalidate all objects for each FTP-over-HTTP request by entering the **ftp-over-http reval-each-request all** global configuration command. In the ACNS 5.3 software release, the **ftp** keyword was replaced with the **ftp-over-http** and **ftp-native** keywords.

Use the **ftp-over-http proxy outgoing host** global configuration command to configure one or more outgoing FTP proxy servers for the Content Engine. Enter the hostname or IP address for the outgoing FTP proxy servers. The primary outgoing FTP proxy server is the parent cache (upstream FTP proxy server) to which you want this Content Engine to direct all of its missed FTP traffic without using ICP or WCCP.

Use the **ftp-over-http proxy anonymous-pswd** global configuration command to specify the password that has to be used during anonymous FTP-over-HTTP operation.

Use the **ftp-over-http proxy active-mode enable** global configuration command to enable active mode on this Content Engine for FTP-over-HTTP mode. In FTP-over-HTTP caching mode, if the **ftp-over-http proxy active-mode** global configuration command is used, the Content Engine first attempts to use active mode with the origin FTP server for the data connection. If the active mode fails, the Content Engine attempts to use passive mode for the data connection.

In FTP-over-HTTP mode, if the **ftp-over-http proxy active-mode** command is not used, the Content Engine first attempts to use passive mode with the FTP server for the data connection and automatically switches to active mode if passive mode is not supported by the FTP server.

Enter the **show ftp-over-http** EXEC command to view the current FTP-over-HTTP configuration on the Content Engine. Enter the **show statistics ftp-over-http** EXEC command to display statistics for the FTP-over-HTTP requests that this Content Engine has handled. For example, the command output shows the number of FTP-over-HTTP requests received by the Content Engine, the number of FTP-over-HTTP hits and misses, as well as the number of FTP-over-HTTP requests that the Content Engine has forwarded to the origin FTP server or to the specified outgoing proxy server. The command output also shows the number of FTP-over-HTTP errors.

To clear FTP-over-HTTP statistics on the Content Engine, enter the **clear statistics ftp-over-http** EXEC command.

Note In the ACNS 5.3 software release, the **show ftp proxy** EXEC command was replaced with the **show ftp-over-http** and **show ftp-native** EXEC commands.

In the ACNS 5.3 software release, the **show statistics ftp** EXEC command was replaced with the **show statistics ftp-over-http** and **show statistics ftp-native** EXEC commands. In the ACNS 5.3 software releases, the **clear statistics ftp** EXEC command was replaced with the **clear statistics ftp-over-http** and **clear statistics ftp-native** EXEC commands.

Designating a Primary Outgoing FTP Proxy Server

In the ACNS 5.2 software and later releases, you can configure up to eight proxy servers for FTP miss traffic (FTP-over-HTTP).

Note At any one time, the Content Engine uses only one of the configured outgoing FTP proxy servers. Proxy servers cannot be used simultaneously.

To configure a Content Engine to direct all FTP-over-HTTP miss traffic to a parent cache without using ICP or WCCP, you must explicitly designate the parent cache as the primary outgoing FTP proxy server for the Content Engine.

Use the **ftp-over-http proxy outgoing host** *host port* **primary** global configuration command to designate a proxy server as the primary outgoing FTP proxy server for the Content Engine, where the following is true:

- *host* is the hostname or IP address of the parent cache (the outgoing FTP proxy server) to which FTP-over-HTTP missed traffic is directed.
- *port* is the port number used by the parent cache to accept missed FTP-over-HTTP requests from the Content Engine.

Use the **primary** keyword to set the specified host as the primary outgoing FTP proxy server. If several servers (hosts) are configured with the **primary** keyword, the last one configured becomes the primary outgoing FTP proxy server for the Content Engine.

In the following example, host 10.1.1.1 on port 8088 is explicitly designated as the primary outgoing FTP proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing FTP proxy server.

```
ContentEngineA(config)# ftp-over-http proxy outgoing host 10.1.1.1 8088 primary
ContentEngineA(config)# ftp-over-http proxy outgoing host 10.1.1.2 220
```

FTP-over-HTTP Proxy Failover

For FTP-over-HTTP proxy caching, there is a primary proxy failover option that you can configure on Content Engines. This feature, which is referred to as the HTTP proxy failover feature, configures the forward proxy server to contact up to eight other proxy servers (outgoing proxy servers) when an FTP-over-HTTP cache miss occurs (when the requested FTP content is not already stored locally in the Content Engine cache).

You can use the **ftp-over-http proxy outgoing** global configuration command to configure up to eight backup Content Engines or any standard proxy servers for the FTP-over-HTTP proxy failover feature. These outgoing proxy servers can be other Content Engines or standard proxy servers that can be contacted to process FTP-over-HTTP cache misses without using ICP or WCCP. The function of these outgoing proxy servers is to process the FTP-over-HTTP cache misses that have been forwarded to them by the forwarding proxy server. One outgoing proxy server functions as the primary server to receive and process all cache miss traffic.

If the primary outgoing proxy server fails to respond to the FTP-over-HTTP request, the server is noted as failed and the requests are redirected to the next outgoing proxy server until one of the proxies services the request.

A failover occurs in the order that the proxy servers were configured. If all of the configured proxy servers fail, the Content Engine can optionally redirect FTP-over-HTTP requests to the origin server specified in the HTTP header if you have used the **ftp-over-http proxy outgoing origin-server** global configuration command. If the **origin-server** option is not enabled, the client receives an error message. Response errors and read errors are returned to the client, because it is not possible to detect whether these errors are generated at the origin server or at the proxy.

Note At any one time, the Content Engine uses only one of the configured outgoing proxy servers. The outgoing proxy servers cannot be used simultaneously. You can view the state of the outgoing FTP-over-HTTP proxy servers in syslog NOTICE messages and with the **show ftp-over-http proxy** EXEC command.

By default, the Content Engine strips the hop-to-hop 407 (Proxy Authentication Required) error code sent by the Internet proxy. If you enter the **ftp-over-http proxy outgoing preserve-407** global configuration command on a Content Engine, the Content Engine sends the 407 error code to the requesting client browser, and the Internet proxy authenticates the client.

Requests with a destination specified in the **proxy-protocols outgoing-proxy exclude** global configuration command bypass the primary outgoing proxy server and the failover proxy servers.

If all of the outgoing proxy servers fail to process the FTP-over-HTTP cache miss, the following occurs:

- If the **ftp-over-http proxy outgoing origin-server** option is enabled, then the Content Engine (forward proxy server) forwards the FTP-over-HTTP cache miss request to the origin server that was specified in the original FTP-over-HTTP request from the client browser.
- If the **ftp-over-http proxy outgoing origin-server** option is not enabled, an error is sent to the requesting client browser. Response errors and read errors are returned to the requesting client browser, because it is not possible to detect whether these errors are generated at the origin server or at the proxy server.

Note In the ACNS 5.1 software and earlier releases, the primary proxy failover feature supported HTTP only, not FTP. In the ACNS 5.2 software and later releases, FTP-over-HTTP support is available.

The **no ftp-over-http proxy outgoing connection-timeout** option causes the timeout to be set to the default value of 300 milliseconds.

In the following example, the Content Engine is configured to redirect FTP-over-HTTP requests directly to the origin server if all of the proxy servers fail:

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

Requests with a destination specified in the **proxy-protocols outgoing-proxy exclude** global configuration command bypass the primary outgoing proxy and the failover proxy servers.

Monitoring Outgoing Proxy Servers and Statistics

A background process on the Content Engine monitors the state of the configured outgoing proxy servers. You can configure the Content Engine to poll the specified outgoing proxy servers at a specific interval in order to monitor their availability.

This monitor interval is the frequency at which the proxy servers are polled. The monitoring interval is specified in seconds and can be from 10 to 300 seconds. The default monitoring interval is 60 seconds. If one of the outgoing proxy servers is unavailable, the polling mechanism waits for the connect timeout (300000 microseconds) before polling the next outgoing proxy server. Use the **ftp-over-http proxy outgoing monitor** command to specify how frequently the Content Engine polls the specified outgoing FTP proxy servers.

In the following example, the Content Engine is configured to monitor the outgoing FTP-over-HTTP proxy servers every 120 seconds:

```
ContentEngine(config)# ftp-over-http proxy outgoing monitor 120
```

You can also monitor outgoing proxy servers by checking the syslog NOTICE messages on the Content Engine.

Examples

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

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

The following example shows how you can verify the changes using the **show ftp-over-http** EXEC command:

```
ContentEngine# show ftp-over-http
FTP heuristic age-multipliers: directory-listing 10% file 10%
Maximum time to live in days: directory-listing 1 file 1
Minimum time to live for all objects in minutes: 1
Incoming Proxy-Mode:
  Configured Proxy mode FTP over HTTP connections on ports: 8080 8081 9090
Outgoing Proxy-Mode:

   Primary Proxy Server:           10.107.193.244 port     21

 Monitor Interval for Outgoing Proxy Servers is 60 seconds

 Timeout period for probing Outgoing Proxy Servers is 300000 microseconds

 Use of Origin Server upon Proxy Failures is disabled.
Active mode of FTP transfer is enabled
Maximum size of a FTP cacheable object is 1 KBytes
Objects are revalidated on each request
```

The following example removes all FTP-over-HTTP proxy ports from the list entered in the previous example even if you specify only a single port number. Ports 8080 and 9090 also cease to remain FTP-over-HTTP proxy ports.

```
ContentEngine(config)# no ftp-over-http proxy incoming 8081
```

The following example disables all the FTP-over-HTTP proxy ports:

```
ContentEngine(config)# no ftp-over-http proxy incoming
```

The following example configures an upstream FTP-over-HTTP proxy with the IP address 172.16.76.76 on port 8888:

```
ContentEngine(config)# ftp-over-http proxy outgoing host 172.16.76.76 8888
```

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

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

The following example configures the maximum size in kilobytes of an FTP object that the Content Engine will cache for FTP-over-HTTP requests. By default, the maximum size of a cacheable object is 2096128 KB.

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

The following example forces the Content Engine to revalidate all objects for every FTP-over-HTTP request:

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

The following example configures a maximum Time To Live of three days in the cache for directory listing objects and file objects for FTP-over-HTTP requests:

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

The following example configures the Content Engine to keep FTP objects in the cache for a minimum of 10 minutes and a maximum of 24 hours (1 day) for FTP-over-HTTP caching:

```
ContentEngine(config)# ftp-over-http min-ttl 10
ContentEngine(config)# ftp-over-http max-ttl hours directory-listing 24 file 24
```

Related Commands

clear statistics ftp-over-http
debug ftp-over-http
ftp-native
show ftp-over-http
show statistics ftp-over-http
wccp ftp-over-http

full-duplex

To configure an interface for full-duplex operation, use the **full-duplex** interface configuration command. To disable this function, use the **no** form of this command.

full-duplex

no full-duplex

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

interface configuration

Usage Guidelines

Use this command to configure an interface for full-duplex operation. Full duplex allows data to travel in both directions at the same time through an interface or a cable. A half-duplex setting ensures that data only travels in one direction at any given time. Although full duplex is faster, the interfaces sometimes cannot operate effectively in this mode. If you encounter excessive collisions or network errors, configure the interface for half duplex rather than full duplex.

Configuring an interface for autosensing causes the full-duplex operation to be disabled. Conversely, configuring an interface for full-duplex operation causes autosensing to be disabled.

When you set the Content Engine Ethernet interface speed or duplex function using the **half-duplex**, **full-duplex**, or **bandwidth** commands, you should turn off the corresponding Ethernet switch port autosense function and set the duplex function and speed manually. If the Ethernet switch port autosense function is turned off, you have to set the Content Engine Ethernet interface duplex function and speed manually to match the Ethernet switch port settings. The Content Engine Ethernet interface **autosense** command will only erase manually set configurations.

Cisco router Ethernet interfaces do not negotiate duplex settings. If the Content Engine is connected to a router directly with a crossover cable, you must manually set the Content Engine interface to match the router interface settings. Disable **autosense** before configuring an Ethernet interface. When **autosense** is on, manual configurations are overridden. You must reboot the Content Engine to start autosensing.

Examples

The following example configures full-duplex operation on FastEthernet interface (slot 1/port 1):

```
ContentEngine# configure
ContentEngine(config)# interface FastEthernet 1/1
ContentEngine(config-if)# full-duplex
```

The following example disables full-duplex operation:

```
ContentEngine(config-if)# no full-duplex
```

Related Commands

half-duplex
interface
show interface
show running-config
show startup-config

gui-server

To enable or specify the number of the Content Engine management graphical user interface (GUI) server port, use the **gui-server** global configuration command. To disable the GUI server port settings, use the **no** form of this command.

gui-server {**enable** | **port** *port* | **secure** {**enable** | **port** *port*}}

no gui-server {**enable** | **port** | **secure** {**enable** | **port**}}

Syntax Description

enable	Enables the graphical user interface.
port	Configures the graphical user interface server port.
port	Port number (1–65535). The default is 8001.
secure	Sets secure access to the graphical user interface.
enable	Enables the secured graphical user interface.
port	Configures the secure graphical user interface server port.
port	Port number (1–65535). The default is 8003 for secured access.

Defaults

Default port for unsecured access: 8001

Default port for secure access: 8003

Command Modes

global configuration

Usage Guidelines

You can configure the Content Engine GUI on a centrally deployed Content Engine for secure or nonsecure access. The secured Content Engine GUI is the default.

Either secure or nonsecure access to the Content Engine GUI is possible but not both. For example, if the secured Content Engine GUI is enabled (for example, https:// access on port 8003), then nonsecure access to the Content Engine GUI (for example, http:// access on port 8001) is not allowed. The port number of the Content Engine GUI is determined when the ACNS software is installed on the Content Engine.

Before logging in to the Content Engine GUI, check that you have the following information:

- Name or IP address of the Content Engine that you want to log in to.
- User account (username and password) that you want to log in with. If you do not have a user account, your ACNS system administrator must create one for you.
- Type of access enabled on the Content Engine GUI (secure or nonsecure).

To access the Content Engine GUI, you must enter the URL or IP address of the Content Engine and the port number. The URL (location) of the Content Engine is determined during the installation of the ACNS software. If your network supports DNS and the IP address of the Content Engine has been added to your DNS table, you can access the Content Engine GUI by using the DNS name of the Content Engine.

Examples

The following example enables the Content Engine management GUI on port 8002:

```
ContentEngine(config)# gui-server enable
ContentEngine(config)# gui-server port 8002
```

Related Commands

show gui-server

half-duplex

To configure an interface for half-duplex operation, use the **half-duplex** interface configuration command. To disable this function, use the **no** form of this command.

half-duplex

no half-duplex

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

interface configuration

Usage Guidelines

Use this command to configure an interface for half-duplex operation. Full duplex allows data to travel in both directions at the same time through an interface or a cable. A half-duplex setting ensures that data only travels in one direction at any given time. Although full duplex is faster, the interfaces sometimes cannot operate effectively in this mode. If you encounter excessive collisions or network errors, configure the interface for half duplex rather than full duplex.

Configuring an interface for autosensing causes the half-duplex operation to be disabled. Conversely, configuring an interface for half-duplex operation causes autosensing to be disabled.

When you set the Content Engine Ethernet interface speed or duplex function using the **half-duplex**, **full-duplex**, or **bandwidth** commands, you should turn off the corresponding Ethernet switch port autosense function and set the duplex function and speed manually. If the Ethernet switch port autosense function is turned off, you have to set the Content Engine Ethernet interface duplex function and speed manually to match the Ethernet switch port settings. The Content Engine Ethernet interface **autosense** command will only erase manually set configurations.

Cisco router Ethernet interfaces do not negotiate duplex settings. If the Content Engine is connected to a router directly with a crossover cable, you must manually set the Content Engine interface to match the router interface settings. Disable **autosense** before configuring an Ethernet interface. When **autosense** is on, manual configurations are overridden. You must reboot the Content Engine to start autosensing.

Examples

The following example configures half-duplex operation on FastEthernet interface (slot 1/port 1):

```
ContentEngine# configure
ContentEngine(config)# interface FastEthernet 1/1
ContentEngine(config-if)# half-duplex
```

The following example disables half-duplex operation:

```
ContentEngine(config-if)# no half-duplex
```

Related Commands

full-duplex
interface
show interface
show running-config
show startup-config

help

To obtain online help for the command-line interface, use the **help** EXEC or global configuration command.

help

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC and global configuration

Usage Guidelines

You can get help at any point in a command by entering a question mark (**?**). If nothing matches, the help list will be empty, and you must back up until entering a **?** shows the available options.

Two styles of help are provided:

- Full help is available when you are ready to enter a command argument (for example, **show ?**). In addition, full help describes each possible argument.
- Partial help is provided when you enter an abbreviated command and you want to know what arguments match the input (for example, **show stat?**).

Examples

The following example shows the output of the **help** EXEC command:

```
ContentEngine# help
Help may be requested at any point in a command by entering a question mark '?'.
Two styles of help are provided:
1. Full help is available when you are ready to enter a command argument.
2. Partial help is provided when an abbreviated argument is entered.
```

hostname

To configure the Content Engine's network hostname, use the **hostname** global configuration command. To reset the hostname to the default setting, use the **no** form of this command.

hostname *name*

no hostname

Syntax Description

name	New hostname for the Content Engine; the name is case sensitive. The name may be from 1 to 30 alphanumeric characters.

Defaults

The default hostname is the Content Engine model number (for example CE590 or CE7320).

Command Modes

global configuration

Usage Guidelines

Use this command to configure the hostname for the Content Engine. The hostname is used for the command prompts and default configuration filenames. This name is also used by content routing and conforms to the following rules:

- It can use only alphanumeric characters and hyphens (-).
- The maximum length is 30 characters.
- The following characters are considered illegal and cannot be used when naming a device: @, #, $,%, ^, &, *, (), |, \””/, <>.

Examples

The following example changes the hostname to Sandbox:

```
ContentEngine(config)# hostname Sandbox
Sandbox(config)#
```

The following example removes the hostname:

```
ContentEngine(config)# no hostname
NO-HOSTNAME(config)#
```

Related Commands

dnslookup
ip
show hosts