

http

To configure HTTP-related parameters, use the **http** global configuration command. To disable HTTP related-parameters, use the **no** form of this command.

http add-method *method*

http age-multiplier *text num binary num*

http always-resolve-host

http anonymizer *enable*

http append { **host-header** | **proxy-auth-header** { *hostname* | *ip-address* } | **via-header** [*custom string*] | **www-auth-header** { *hostname* | *ip-address* } | **x-forwarded-for-header** }

http authenticate-strip-ntlm

http authentication { **cache** { **max-entries** *entries* | **max-group-entries** *number* | **timeout** *minutes* | **ttl** *minutes* } | **header** { **401** | **407** } | **realm** *line* }

http cache-authenticated { **all** | **basic** | **ntlm** }

http cache-chunk-encoded *enable*

http cache-cookies

http cache-on-abort { **enable** | **max-threshold** *maxthresh* | **min-threshold** *minthresh* | **percent** *percentthresh* }

http cache-vary-user-agent { **enable** | **sub-string** *user_agent_string* **cache-string** *string* }

http client-no-cache-request { **ignore** | **revalidate** }

http cluster { **heal-port** *number* | **http-port** *number* | **max-delay** *seconds* | **misses** *number* }

http destination-port { **allow** { **all** | *range* } | **deny** { **all** | *range* } }

http dns-cache { **serial-lookup** | **size** *max-num* }

http fast-response *enable*

http l4-switch { **enable** | **spoof-client-ip** *enable* }

http max-ttl { **days** *text textdays* **binary** *bindays* | **hours** *text texthours* **binary** *binhours* | **minutes** *text textminutes* **binary** *binminutes* | **seconds** *text textseconds* **binary** *binseconds* }

http min-ttl *minutes*

http monitor url *url* [**acceptable-delay** *seconds* [**interval** *seconds*] | **interval** *seconds* [**acceptable-delay** *seconds*]]

http object { **max-size** *maxsize* | **url-validation** *enable* }

http persistent-connections [**all** | **client-only** | **server-only** | **timeout** *seconds*]

http proxy {**incoming** *ports* | **outgoing** {**connection-timeout** *microsecs* | **host** {*hostname* | *ip-address*} *port* [**primary**] | **monitor** *seconds* | **origin-server** | **preserve-407**}}

http request-header **host** **unmodified**

http reval-each-request {**all** | **none** | **text**}

http serve-ims **text** *percentage* **binary** *percentage*

http smart-range {**enable** | **max-start** *offset* **max-interval** *interval*}

http strict-request-content-length-checking **enable**

http tcp-keepalive **enable**

no http {**add-method** *method* | **age-multiplier** **text** *num* **binary** *num* | **always-resolve-host** | **anonymizer** **enable** | **append** {**host-header** | **proxy-auth-header** {*hostname* | *ip-address*} | **via-header** | **www-auth-header** {*hostname* | *ip-address*} | **x-forwarded-for-header** | **authenticate-strip-ntlm** | **authentication** {**cache** {**max-entries** | **max-group-entries** | **timeout** | **ttl**} | **header** | **realm**} | **cache-authenticated** {**all** | **basic** | **ntlm**} | **cache-chunk-encoded** **enable** | **cache-cookies** | **cache-on-abort** {**enable** | **max-threshold** *maxthresh* | **min-threshold** *minthresh* | **percent** *percentthresh*} | **cache-vary-user-agent** {**enable** | **sub-string** *user_agent_string* **cache-string** *string*} | **client-no-cache-request** | **cluster** {**heal-port** *number* | **http-port** | **max-delay** | **misses**} | **destination-port** {**allow** | **deny** {**all** | *range*}} | **dns-cache** [*size*] | **fast-response** | **l4-switch** {**enable** | **spoof-client-ip** **enable**} | **max-ttl** | **min-ttl** | **monitor** *url* *url* [**acceptable-delay** *seconds* [*interval* *seconds*] | **interval** *seconds* [*acceptable-delay* *seconds*]] | **object** {**max-size** | **url-validation** **enable**} | **persistent-connections** [**all** | **client-only** | **server-only** | **timeout**] | **proxy** {**incoming** | **outgoing** {**connection-timeout** | **host** {*hostname* | *ip-address*} *port* [**primary**] | **monitor** | **origin-server** | **preserve-407**}} | **request-header** **host** **unmodified** | **reval-each-request** | **serve-ims** | **smart-range** [**enable**] | **strict-request-content-length-checking** **enable** | **tcp-keepalive**}

Syntax Description

add-method	Adds the HTTP request method.
<i>method</i>	Method to be added to the list of request methods supported by the Content Engine.
age-multiplier	Specifies the HTTP/1.0 caching heuristic modifiers.
text	Specifies the heuristic modifier for the text object.
<i>num</i>	Expiration time of text objects as a percentage of their age (0–100).
binary	Specifies the heuristic modifier for the binary object.
<i>num</i>	Expiration time of binary objects as a percentage of their age (0–100).
always-resolve-host	Sets the Content Engine to always resolve the Host header to determine the origin server's IP address.
anonymizer enable	Sets the HTTP anonymizer.
append	Configures the HTTP headers to be appended by the Content Engine. You can configure the Content Engine to append a particular header to either the request or response headers or both depending on the header selected.

host-header	Appends the host headers to HTTP/1.0 requests that do not possess these headers.
proxy-auth-header	Configures the host to receive the Proxy Authorization header if it is present in the request. The append proxy-auth-header must be configured on the downstream Content Engines to ensure that proxy authorization information, required by upstream Content Engines, is not stripped from the HTTP request by the downstream Content Engines. Up to eight upstream IP addresses can be configured on each downstream Content Engine.
<i>hostname</i>	Name of host receiving the Proxy Authorization header.
<i>ip-address</i>	IP address of host receiving the Proxy Authorization header.
via-header	Appends the HTTP request or response version number to an existing Via request or response header, or appends a new Via header if there is no header in the HTTP request.
custom	(Optional) Configures a custom string to be displayed in the Via header description.
<i>string</i>	String that forms the custom Via header. Supports up to a maximum of 99 characters. Valid characters are alphanumeric characters and the following special characters: semicolon (;), colon (:), period (.), and white spaces. Note The Via header and custom Via header are mutually exclusive, and the last selection takes effect.
www-auth-header	Configures the host to receive the Authorization request header when that header was intended for use by this Content Engine for HTTP proxy request authentication.
<i>hostname</i>	Name of host receiving WWW Authorization header.
<i>ip-address</i>	IP address of host receiving WWW Authorization header.
x-forwarded-for-header	Notifies the web server of the client's IP address through the X-Forwarded-For header. Allows the Content Engine to append an X-Forwarded-For request header that contains the client's IP address when there is no X-Forwarded-For header present in the request.
authenticate-strip-ntlm	Strips the NT LAN Manager (NTLM) authentication headers.
authentication	Configures the parameters related to HTTP authentication.
cache	Configures the parameters that are related to the authentication cache on the Content Engine.
max-entries	Sets the maximum number of entries retained in the authentication cache.
<i>entries</i>	Maximum number of entries retained in the authentication cache on the Content Engine. Valid values are from 500–32000 entries. This number is limited by the physical resources available on the Content Engine.

max-group-entries	Sets the maximum number of entries retained in the authentication group cache. This number is subject to physical resources on the Content Engine. This option is only available in the ACNS software Release 5.2 and later releases. Tip If you intend to use the group name pattern, make sure that you set the correct number of maximum group entries in the authentication group cache. This number should correspond to the maximum number of groups that could be returned during authorization queries (for example, the total number of groups defined on the AAA server).
<i>number</i>	Maximum number of entries retained in the authentication group cache on the Content Engine. Valid values are from 500–12000 entries. This number is limited by the physical resources available on the Content Engine.
timeout	Sets the timeout value of records in the authentication cache. Specifies how long an inactive entry can remain in the authentication cache before it is purged. Once a record has been purged, any subsequent access attempt to restricted Internet content requires a server lookup for reauthentication. This keyword is the least-recently-used (LRU) value. It is also referred to as the idle time.
<i>minutes</i>	Time in minutes (1–1440) between the user's last Internet access and the removal of that user's entry from the authorization cache, forcing reauthentication. The default is 240 minutes (4 hours), the minimum is 30 minutes, and the maximum is 1440 minutes (24 hours).
ttl	Sets an absolute Time To Live (TTL) for entries in the authentication cache. This option is only available in the ACNS software Release 5.2 and later releases. By default, this option is disabled, which means that there is no TTL timeout in effect. This default means that there will be no check to time out an authentication cache entry based on its creation time relative to a TTL value.
<i>minutes</i>	Time in minutes (1–1440) that specifies the maximum amount of time that an entry is valid in the authentication cache entry after its creation. The minimum is 1 minute; the maximum is 1440 minutes (24 hours). For more information, see the “Specifying a Reauthentication Interval” section on page 2-211.
header	Determines which HTTP header to use for authentication (user ID and password) when the style of the HTTP request indicates that no proxy server is present. Headers can be either HTTP 401 (Unauthorized) or HTTP 407 (Proxy Authentication Required). The default is HTTP 401.
401	Uses HTTP 401 to query users for credentials.
407	Uses HTTP 407 to query users for credentials.
<i>realm</i>	Realm string for basic HTTP request authentication.
<i>line</i>	Name of the realm string to be authenticated.
cache-authenticated	Caches and revalidates the authenticated web objects.
all	Caches the web objects that were authenticated with any authentication scheme (basic authentication or NTLM authentication).

basic	Caches the web objects that were authenticated using the basic scheme authorization.
Note	Basic authentication is less secure than NTLM authentication because it transmits the user credential information in clear text format.
ntlm	Caches the web objects that were authenticated using the NTLM scheme authorization.
cache-chunk-encoded	Caches the chunked transfer-encoded (CTE) HTTP objects. For more information, see the “Chunked Transfer Coding” section on page 2-213 .
enable	Enable the caching of the chunked transfer-encoded HTTP objects.
cache-cookies	Caches the web objects with associated cookies.
cache-on-abort	Sets the cache-on-abort configuration options.
enable	Enables the cache-on-abort feature.
max-threshold	Sets the maximum threshold for cache on abort.
<i>maxthresh</i>	Value in kilobytes of maximum threshold (1–99999). The default is 256.
min-threshold	Sets the minimum threshold for cache on abort.
<i>minthresh</i>	Value in kilobytes of minimum threshold (1–99999). The default is 32.
percent	Sets the percent threshold.
<i>percentthresh</i>	Percentage value (1–99). The default is 80 percent.
cache-vary-user-agent	Configures the parameters for caching responses with the Vary header.
enable	Enables the caching objects with the Vary header.
sub-string	Configures the string to be used to match while searching for an alternative user-agent.
<i>user-agent-string</i>	String to be used while matching an alternative user agent.
cache-string	Configures the cache-string that forms the objects key.
<i>string</i>	String that forms the objects key. The maximum is 16 characters.
client-no-cache-request	Configures the management of no-cache requests.
ignore	Ignores the no-cache header in a client request.
revalidate	Revalidates the object with the origin server before serving a no-cache client request.
cluster	Configures the cache cluster options.
heal-port	Specifies the listening port number of the healing server for healing requests.
<i>number</i>	Healing server listener port number (1–65535). The default is 14333.
http-port	Specifies the healing server HTTP request forwarding port number.
<i>number</i>	HTTP request forwarding port number (1–65535). The default is 80.
max-delay	Specifies the maximum wait for response.
<i>seconds</i>	Maximum delay in seconds (0–10).
misses	Specifies the duration of the healing mode (misses).
<i>number</i>	Total number of misses (0–999) before the healing mode is disabled.

destination-port	Sets the destination port restrictions.
allow	Allows the HTTP traffic to specified ports. The default is to allow traffic to port range 80–87 and to any port above 1024.
all	Specifies all ports.
<i>range</i>	Port numbers (1–65535). You can configure a maximum of eight ports.
deny	Denies the HTTP traffic to the specified ports. Port numbers below port 1024 are denied by default, except for port range 80–87.
all	Specifies all ports.
<i>range</i>	Port numbers (1–65535). Up to eight port ranges can be configured.
dns-cache	Configures the web proxy DNS cache.
serial-lookup	Queries each of the configured name servers in turn if the primary server responds in the negative.
size	Sets the DNS cache size.
<i>max-num</i>	Maximum number of cache records (4096–65536).
fast-response	Configures the fast HTTP client response.
enable	Enables the fast HTTP client response.
l4-switch	Configures parameters for Layer 4 switch redirection.
enable	Enables the Content Engine to serve requests redirected by a Layer 4 switch, such as a Cisco Content Switching module or Content Services switch.
spoof-client-ip	Bypasses the packets unmodified so that they always contain the IP address of the Content Engine.
enable	Enables the client IP spoofing.
max-ttl	Sets the maximum Time To Live for objects in the cache.
days	Sets the maximum Time To Live for units in days.
text	Sets the maximum Time To Live for text objects.
<i>textdays</i>	Maximum Time To Live (1–1825). The default is three days.
binary	Sets the maximum Time To Live for binary objects.
<i>bindays</i>	Maximum Time To Live (1–1825). The default is seven days.
hours	Sets the maximum Time To Live for units in hours.
<i>texthours</i>	Maximum Time To Live (1–43800). The default is 72 hours.
<i>binhours</i>	Maximum Time To Live (1–43800). The default is 168 hours.
minutes	Sets the maximum Time To Live for units in minutes.
<i>textminutes</i>	Maximum Time To Live (1–2628000). The default is 4320 minutes.
<i>binminutes</i>	Maximum Time To Live (1–2628000). The default is 10080 minutes.
seconds	Sets the maximum Time To Live for units in seconds.
<i>textseconds</i>	Maximum Time To Live (1–157680000). The default is 259200 seconds.
<i>binseconds</i>	Maximum Time To Live (1–157680000). The default is 604800 seconds.
min-ttl	Sets the minimum Time To Live for objects in the cache.
<i>minutes</i>	Minimum Time To Live in minutes (0–86400).

monitor	Configures the Content Engine to monitor the performance of a specific URL.
url	Specifies the URL to be monitored.
<i>url</i>	URL to be monitored.
acceptable-delay	(Optional) Sets the maximum number of seconds that the specified monitored URL should be retrieved within.
<i>seconds</i>	Acceptable delay in seconds (1–3600). The default is 60 seconds.
interval	(Optional) Sets the frequency at which the Content Engine should monitor requests for a specific URL.
<i>seconds</i>	Monitoring interval in seconds (1–3600). The default is 60 seconds.
object	Sets the URL validation and maximum size of HTTP objects.
max-size	Sets the maximum size of a cacheable object.
<i>maxsize</i>	Maximum size of a cacheable object in kilobytes (1–2096128).
url-validation enable	Enables each HTTP validation request.
persistent-connections	Sets the configuration options for the persistent connections.
all	(Optional) Makes the client and server connections persistent.
client-only	(Optional) Makes only a client connection persistent.
server-only	(Optional) Makes only a server connection persistent.
timeout	(Optional) Sets the timeout value for the persistent connections.
<i>seconds</i>	Timeout in seconds (1–86400) for the persistent connections. The default is 600 seconds.
proxy	Configures the parameters for proxy mode.
incoming	Configures for incoming proxy-mode requests.
<i>ports</i>	Ports on which to listen for incoming HTTP, FTP, and HTTPS proxy requests (1–65535). You can specify a maximum of eight ports. The default is no incoming proxy.
outgoing	Configures the direct outgoing requests to another proxy server.
connection-timeout	Defines a timeout period, in microseconds, for probing outgoing proxy servers.
<i>microsecs</i>	Timeout period in microseconds (200–5000000).
host	Uses the outgoing HTTP proxy.
<i>hostname</i>	Hostname of the outgoing proxy.
<i>ip-address</i>	IP address of the outgoing proxy.
<i>port</i>	Port number of the outgoing proxy (1–65535).
primary	(Optional) Makes the configured proxy as the primary proxy server.
monitor	Defines the interval for monitoring the outgoing proxy servers.
<i>seconds</i>	Monitoring interval in seconds (10–300).
origin-server	Uses the origin server if all outgoing proxies fail.
preserve-407	Preserves the HTTP authentication header 407 by sending header 407 to the client when asking the client for Internet proxy authentication credentials.
request-header host unmodified	Specifies whether or not to append a default domain name to a hostname string even if this string does not contain a fully qualified domain name (FQDN).

reval-each-request	Configures the revalidation for every request.
all	Revalidates all objects on every request.
none	Does not revalidate objects for each request.
text	Revalidates the text objects on every request.
serve-ims	Configures the if-modified-since (IMS) requests.
text	Modifies the IMS request handling of text objects.
<i>percentage</i>	Age percentage to serve a text object without revalidation.
binary	Modifies the IMS request handling of binary objects.
<i>percentage</i>	Age percentage to serve a binary object without revalidation.
smart-range	Manages the Range requests if a cache miss occurs.
enable	Enables the smart Range request caching feature.
max-start	Specifies the maximum starting offset of the range request to be cached.
<i>offset</i>	Value of starting offset in bytes (0–2147483647). The default is 16384.
max-interval	Specifies the maximum interval between any subrange in the range request.
<i>interval</i>	Value of maximum interval in bytes (0–2147483647). The default is 16384.
strict-request-content-length-checking	Enables the checking options for the request content length.
enable	Enables the strict request content length checks.
tcp-keepalive	Configures the TCP keepalive feature.
enable	Enables the TCP keepalive on HTTP connections.

Defaults

age-multiplier: 30 percent for text objects and 60 percent for binary objects

timeout *minutes*: 480

header: HTTP 401

maxthresh: 256 KB

minthresh: 32 KB

ports: no incoming proxy

percentthresh: 80 percent

heal-port *number*: 14333

http-port *number*: 80

texdays: 3 days

bindays: 7 days

texthours: 72 hours

binhours: 168 hours

textminutes: 4320 minutes

binminutes: 10080 minutes

textseconds: 259200 seconds

binseconds: 604800 seconds

misses number: 0 misses

monitor url url acceptable-delay: 60 seconds

monitor url url interval: 60 seconds

object max-size: 2 GB

outgoing connection-timeout: 300 milliseconds

outgoing monitor: 60 seconds

http strict-request-content-length-checking: disabled

The Content Engine strips the hop-to-hop 407 response sent by the Internet proxy by default.

http cache-on-abort: disabled

http destination-port: allows ports 80–87 and ports 1025–65535; denies ports 1–79 and 88–1024

max-start offset: 16384 bytes

max-interval interval: 16384 bytes

The default is no incoming proxy.

http tcp-keepalive: disabled

Command Modes

global configuration

Usage Guidelines

Use these commands to configure specific parameters for caching HTTP objects.

The **http anonymizer** command zeros out client IP addresses in the log files.



Note

Text objects refer to HTML pages. Binary objects refer to all other web objects (for example, GIFs or JPEGs).



Note

For information on http cache freshness, see Chapter 7 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

http add-method Command Considerations

In the ACNS software, HTTP request methods are categorized as supported and unsupported. For instance, Content Engines accept or reject HTTP requests depending on whether or not the request method is supported. The request method in an incoming request from a client is matched against the list of supported methods. If the method is found in the list of supported methods, the request is processed by the Content Engine; if the method is not found, the request is not served. HTTP 1.1 request methods (for example, GET, HEAD, or POST) are supported by default. Nonstandard request methods, such as Web-Based Distributed Authoring and Versioning (WebDAV), are not supported.

You can use the **http add-method** global configuration command to let you add HTTP request methods to the list of those supported by the Content Engine. Use the **no http add-method** command to remove a method from the list.

You can run the **show http methods EXEC** command to display a list of supported and unsupported HTTP request methods. In the **show http methods** command output, request methods supported by default appear in parentheses.

REPORT is listed as the last entry in the output. It does not appear in parentheses, because REPORT is not a default method.

A description of WebDAV and WebDAV methods is available as IETF RFC 2518, *HTTP Extensions for Distributed Authoring—WEBDAV*.

**Note**

When the Content Engine receives an HTTP request from a client using a method not supported, the ACNS software adds the method to the list of unsupported methods and returns an error to the client. You can add any method not supported to the list of supported methods.

HTTP Request Considerations

The ACNS 5.x software caching services support Microsoft NT LAN Manager (NTLM), Lightweight Directory Access Protocol (LDAP), and RADIUS server HTTP request authentication. The **http authentication** command authenticates a user's domain, username, and password with a preconfigured primary domain controller (PDC) before allowing requests from the user to be served by the Content Engine.

When the access control list is configured and enabled, an NTLM or LDAP authenticated user has to belong to an access control list to allow access to the requested content. However, even with the access control list enabled, the default policy is to allow access to the requested content, which means that if the user does not appear in any access control lists, access is allowed.

**Note**

The ACNS 5.x software only allows group authorization using access control lists for users who have been authenticated using either an NTLM or an LDAP server for HTTP requests.

**Note**

All authentication schemes using NTLM, TACACS+, LDAP, and RADIUS servers, which may require different user IDs and passwords, are mutually exclusive; only one authentication scheme can be enabled at a time.

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

The maximum number of entries that is maintained in the authentication cache is 32,000. The minimum number is 500. The default value is 16,000 entries. Use the **http authentication max-entries** global configuration command to configure the maximum number of entries that is to be maintained in the authentication cache on this Content Engine, if necessary.

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

The **http authentication realm** global configuration command configures the realm displayed in the authentication popup window. When you connect to the Content Engine, this realm is “Cisco Content Engine.” Because this realm reflects the type of application or device being used, an option to configure the realm to be more generic was provided, such as “proxy” or “cache,” so that a realm string can be configured for HTTP basic request authentication.

The **show http authentication EXEC** command can be used to view the name of the configured realm.

Specifying a Reauthentication Interval

In the ACNS 5.1 software, an inactivity timer was used to determine when the client browser was prompted to reenter authentication credentials after being initially authenticated. This inactivity timer is configured with the **http authentication cache timeout** global configuration command. By default, the inactivity timeout period was 8 hours. As long as someone continues to use the client browser after the legitimate authenticated user was initially authenticated, the client is not forced to reenter that person’s authentication credentials.

The ability to specify an absolute TTL for HTTP authentication cache entries was added in the ACNS 5.2 software for increased security in a shared workstation environment. This ability to specify an absolute TTL timeout adds security to the inactivity timeout mechanism for the content that is served through the Content Engine. When the absolute TTL period has expired, the client browser is forced to reauthenticate itself, and the user must enter valid credentials.

A security risk exists in a shared workstation environment that is using WCCP-enabled router redirection mode with any authentication method or proxy redirection mode and the NTLM authentication method. In these cases, the Content Engine uses the client’s IP address as the index into the authentication record kept in the authentication cache, and the Content Engine can authenticate users who have not presented valid credentials of their own if a different user using the same workstation has previously presented valid credentials that are cached in the authentication cache. To provide additional security, you can configure the absolute TTL for HTTP authentication cache entries; this absolute TTL specifies the absolute time during which an authentication cache entry is valid in the cache. If a cache lookup occurs on an entry and its configured TTL time is exceeded, then the entry is deleted and the Content Engine queries the user for credentials.

To use the absolute TTL in a shared workstation environment that uses the Internet Explorer browser, there are additional considerations. By default, the browser automatically sends the workstation logon credentials when the Content Engine queries the user for credentials. So that the absolute TTL can provide additional security, either the logon credentials must not be valid request authentication credentials for the Content Engine or the Internet Explorer browser must configure its security settings to not send the logon credentials automatically when the Content Engine queries the user for credentials. To configure this feature on the Internet Explorer browser, choose **Tools > Internet Options > Security > Internet (and/or the other zones) > Custom Level > User Authentication > Logon > Prompt for Username and Password**. For this browser configuration to be effective, it must receive a 401 HTTP reply code when it is queried for credentials, which is the default in transparent redirection using WCCP. To use the 401 HTTP reply code when you are in proxy redirection mode, use the Content Engine **http authentication header 401** global configuration command. To support this feature, the **ttl** option was added to the **http authentication cache** global configuration command.

TACACS+ Request Authentication

The TACACS+ database also validates users during an HTTP request authentication. TACACS+ provides both authentication and authorization options. To configure TACACS+, use the **authentication** and **tacacs** commands. To enable TACACS+, use the **tacacs enable** command.

For more information on TACACS+ authentication, see the [“tacacs” section on page 2-761](#).

NTLM HTTP Request Authentication

The NTLM protocol can be used to authenticate and block user access to the Internet. When a user logs in to a Windows NT or a Windows 2000 domain and starts a browser, the authentication information is stored by the browser and later used as NTLM credentials to access the Internet. The browser sends the NTLM credentials with the domain name to the ACNS cache, which in turn sends a request to the Windows NT domain controller to check the validity of the user in the domain. If the user is not a valid user in the domain, then the request to access the Internet is denied. If authentication succeeds, the source IP address is entered in the authentication cache. Future requests from this IP address are not challenged until the authentication cache entry expires or is cleared. For more information on NTLM authentication, see the “[ntlm](#)” section on page 2-344.



Note

In the case of HTTP request authentication, the ACNS 5.x software supports only NTLM Version 1.

RADIUS HTTP Request Authentication

RADIUS authentication clients reside on the Content Engine running the ACNS 5.x software. When enabled, these clients send authentication requests to a central RADIUS server, which contains user authentication and network service access information.

To configure RADIUS parameters, use the **radius-server** command in global configuration mode. To disable RADIUS authentication parameters, use the **no** form of this command. For more information on RADIUS authentication, see the “[radius-server](#)” section on page 2-384.

LDAP HTTP Request Authentication

System administrators can use the Content Engine to restrict user Internet access using an LDAP server for authentication purposes, which provides most of the services of the X.500 protocol with less complexity and overhead.

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

The ACNS 5.x software supports LDAP Version 2 and Version 3 and supports all LDAP features except for Secure Authentication and Security Layer (SASL). For more information on LDAP authentication, see the “[ldap](#)” section on page 2-291.

Excluding Domains from HTTP Authentication Servers

To exclude domains from HTTP authentication servers, use the **rule action no-auth pattern-list** *number* command. TACACS+, NTLM, RADIUS, or LDAP authentication takes place only if the site requested does not match the specified domain from the pattern list chosen. For more information on excluding domains using **rule** commands, see the “[rule](#)” section on page 2-407.



Note

For information on proxy mode proxy server authentication and transparent mode authentication, see Chapter 10 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

End-to-End Authentication

The ACNS 5.x software caching services support both basic and NTLM end-to-end authentication. End-to-end NTLM authentication includes pass-through servicing and the caching of web objects that require NTLM authentication. HTTP request authentication authenticates a user's domain, username,

and password with a preconfigured NTLM domain controller before allowing requests from the user to be served by the Content Engine. NTLM authentication works only in a Microsoft environment (for instance, Microsoft Internet Explorer clients accessing Microsoft Internet Information Servers).

**Note**

The ACNS 5.x software supports NTLM Version 1 and Version 2 for end-to-end authentication.

**Note**

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

Basic End-to-End Authentication

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

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

**Note**

For information on NTLM end-to-end authentication, see Chapter 10 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

**Note**

For information on hierarchical caching in proxy mode, see Chapter 10 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

Chunked Transfer Coding

The HTTP 1.1 specification allows objects to be transmitted using Chunked Transfer Coding. The ACNS software, Release 5.1 and earlier releases, supported the proxying of chunked transfer encoded (CTE) objects was supported; however, the caching of these objects was not supported.

The ACNS software, Release 5.2 and later releases, support the caching of CTE HTTP objects. A subsequent request for the same CTE object, which meets the standard HTTP caching freshness criteria, results in that object being fetched from the Content Engine's cache and sent to the client that is HTTP 1.1 compliant. HTTP 1.0 clients do not support CTE. If an object is stored in a CTE format, then the Content Engine must refetch the object from the source HTTP server if the HTTP client is not HTTP 1.1 compliant.

To enable the caching of CTE HTTP objects on a Content Engine, specify the **http cache-chunk-encoded enable** global configuration command. After enabling this feature, you can use the **show statistics http request EXEC** command to verify that this feature is working properly. Check the command output to verify whether the displayed value in the Chunked HTTP Responses: field increments after the Content Engine serves cached CTE HTTP objects to HTTP 1.1-compliant clients.

**Note**

For details on HTTP, see the IETF RFC 1945 (HTTP 1.0 specification) and RFC 2616 (HTTP 1.1 specification).

Security Options

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

To prevent unwanted access to unexpected HTTP ports above 1024 when a request is going through the Content Engine, use the **http destination-port deny 1025-65535** command. This command sequence denies access to any port except for ports 80–87.

To allow HTTP access to a specific port number below 1025, use the **http destination-port allow 88** command.

Hierarchical Caching in Transparent Mode

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

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

**Note**

For information on hierarchical caching in transparent mode, see Chapter 10 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

Authentication Cache Size Adjustments

If the authentication cache is not large enough to accommodate all authenticated users at the same time, the Content Engine purges older entries that have not timed out.

Use the **http authentication cache timeout** command to configure the authentication cache timeout parameters if necessary.

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

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

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

Enabling the Caching of Authenticated Content

By default, the authenticated content is not cached in HTTP. You can use the **http cache-authenticated** global configuration command to change this default policy.

When NTLM object caching is enabled on a Content Engine and the Content Engine receives an NTLM HTTP request from a client, it searches its cache for the requested content. If a cache hit occurs, the Content Engine sends the requested content to the client. If a cache miss occurs, the Content Engine retrieves the requested content from the origin server, caches a local copy of the content, and sends the requested content to the client.

The cached objects are tagged as “NTLM protected” so that subsequent requests for these same objects are subjected to authentication before the Content Engine can serve the content to the client. If there is a cache hit and the requested object is an NTLM protected object, the Content Engine checks whether there is a secure connection between this client and the server as follows:

- If there is a secure connection, the Content Engine sends an if-modified-since (IMS) request to the server using the proxy server connection.
- If the Content Engine receives a 304 response from the server, the Content Engine serves the cached content to the client.
- If the Content Engine receives a 200 response from the server, the Content Engine caches the new object and serves it to the client.
- If there is no established secure connection between the client and the server, the Content Engine attempts to establish the secure connection using IMS messages.

Transaction Logging

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

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

Use the **no ldap server** command to delete an LDAP server.

The **cache-cookies** option enables the Content Engine to cache the binary content served with HTTP Set-cookie headers and no explicit expiration information.

The **reval-each-request** option enables the Content Engine to revalidate all objects requested from the cache, text objects only, or none at all.

Use the **object max-size** option to specify the maximum size in kilobytes of a cacheable object. The default is no maximum size for a cacheable object. The **no** form of the command resets the default value.

The **http proxy** options enable the Content Engine to operate in environments where WCCP is not enabled, or where client browsers have previously been configured to use a legacy proxy server. The Content Engine accepts proxy-style requests when the incoming proxy ports are configured with the **http proxy incoming ports** option. Up to eight incoming proxy ports can be specified on a single command line or on multiple command lines.

To configure the Content Engine to direct all HTTP miss traffic to a parent cache (without using ICP or WCCP), use the **http proxy outgoing host port** option, where **host** is the system name or IP address of the outgoing proxy server, and **port** is the port number designated by the outgoing (upstream) server to accept proxy requests.

Caching Policy for Client-Aborted Downloads

Typically, a client aborts a download of an object by clicking the **Stop** icon on the browser or by closing the browser during a download. By default, the Content Engine continues to download an object to the cache even after a client aborts the download.

The **cache-on-abort** option lets you specify if and when the Content Engine completes the download of a cacheable object after the client aborts the request. However, if the Content Engine determines that there is another client currently requesting the same object, caching is always completed.

If the **cache-on-abort** option is enabled and no thresholds are enabled, the Content Engine always aborts downloading an object to the cache. You can use any combination of the following thresholds, which are applied in the order shown:

- **min-threshold**—The Content Engine continues to cache an object only if the number of kilobytes (KB) remaining to be downloaded from the server is less than or equal to the value of *minthresh*. The default is 32 KB.
- **max-threshold**—The Content Engine continues to cache an object only if the number of kilobytes remaining to be downloaded from the server is less than or equal to the value of *maxthresh*. The default is 256 KB.
- **percent**—The Content Engine continues to cache an object if the percentage downloaded is greater than *percentthresh*. The default is 80 percent.

Other Options

The **client-no-cache-request** option allows a choice between ignoring the no-cache client request or revalidating the object with the origin server before serving the no-cache client request. These choices are mutually exclusive, and the last selection takes effect.

The **fast-response enable** command disables Nagle's algorithm for all HTTP connections. This command should be used only if there are current applications running that require an immediate, timely response to frequent small bursts of information from clients. Disabling Nagle's algorithm can result in performance degradation because it disables the buffering of data. This algorithm should only be disabled when a particular server application needs the client's information immediately after it is sent. The decision to use this command depends upon the requirements of the client/server applications that are currently running.

Configuring the DNS Cache

DNS caching allows the Content Engine (in proxy and WCCP mode) to cache DNS entries to avoid multiple WAN accesses for DNS server resolution. The DNS cache size is the maximum number of DNS cache entries. You can use the **http dns-cache size** command to configure the maximum number of records allowed in the DNS cache.

You can use the **http dns-cache serial-lookup** command to query all configured name servers if the initial DNS server that was contacted fails to respond.



Note

The domain name resolution requires that at least one DNS name server be configured with the **ip name-server** global configuration command. The DNS cache goes online when the **ip name-server** command is configured and goes offline when the last IP name server configuration is deleted with the **no ip name-server ip-address** command.

Transparent Redirection with Layer 4 Switches

Use the **http l4-switch enable** option to allow the Content Engine to transparently receive Layer 4 redirected traffic from Layer 4-enabled switches such as a Cisco Content Switching module or Content Services switch. See the switch documentation for specific configuration information.

**Note**

The **http i4-switch** command enables transparent redirection on HTTP port 80 only. If you want to intercept traffic on a different port, you must configure a WCCP service without a router, that contains the alternative port that you want to use.

If the version of ACNS that you are using does not accept the **wccp** command without the **router-list num** option, you can use a dummy router address. The Content Engine will accept redirected traffic and will send WCCP announcements to the configured router. You can avoid having the Content Engine send WCCP announcements to the configured router by later removing the dummy router list from the configured WCCP service; however, this configuration will be lost after you reload.

Use the **http i4-switch spoof-client-ip enable** command to allow the Content Engine to use the client IP address when redirecting requests through a Layer 4 switch to an origin server.

When enabling redirection with a Layer 4 switch, you need to identify the router to which the Content Engine can direct responses when errors are received from the origin server. Use the **bypass gateway ipaddress** command and replace the *ipaddress* with the IP address of a router that is a Layer 2 neighbor of the Content Engine.

Configuring Healing Mode

When a Content Engine is added to an existing WCCP Version 2 cache group (cluster), it can receive requests for content that were formerly served by another cache in the cluster. This event is termed a near-miss, because if the request had been sent to the former Content Engine, it would have been a cache hit. A near-miss lowers the overall cache hit rate of the Content Engine cluster.

Healing mode allows the newly added Content Engine to query and obtain cache objects from all other caches in the cluster on a cache miss event. If the object is not found in the cluster, the Content Engine processes the request through the outgoing proxy or origin server. The Content Engine in healing mode is called a healing client. The caches in the cluster that respond to healing client requests are called healing servers.

**Note**

Healing mode is only invoked on a healing client when the request is transparently redirected to the Content Engine. Healing mode is not invoked when the request is sent to the Content Engine in proxy mode.

The **http cluster** command modifies the healing mode parameters. The **http cluster http-port** command specifies the port number over which requests from the healing Content Engine are sent to other Content Engines in the cluster.

**Note**

The default port number is 80. If you choose to configure a port other than 80, you must make sure that the port that is configured matches the port specified in the **http proxy incoming** command on healing servers in the farm. Otherwise, the healing client is not able to retrieve objects from the healing servers.

To return to the default port number, use the **no http cluster http-port** command.

The **http cluster heal-port** command specifies the port number over which the healing client sends healing queries and the healing server sends healing responses. The default port number is 14333. If a port other than the default is configured, make sure that all Content Engines in the cluster use the same port.

The **http cluster misses** command specifies the maximum number of misses that the healing Content Engine can receive from the cluster from the last healing mode hit response until the healing process is disabled. The default is 0 misses. The **http cluster max-delay** command specifies the maximum time in seconds that a healing Content Engine waits for a healing response from the cluster before considering that the healing request is a miss.

To enable the healing client, you should, at the least, configure the **max-delay** and **misses** options. The default port number for the **http-port** option is 80. If you use the default port, you do not have to configure the **http-port** option. The default port number for the **heal-port** option is 14333.

**Note**

Pre-positioned content requests are only accepted and served on the HTTP default port, number 80. If the default HTTP port number is changed to any other port number on the Content Engine when the Content Engine is waiting for the pre-positioned content from the origin server, the Content Engine is unable to serve the content.

To disable the healing client, you should, at the least, configure either **misses** or **max-delay** to 0, or you can use the **no** form of the command as follows:

http cluster misses 0

no http cluster misses

http max-delay 0

no http cluster max-delay

**Note**

Healing mode existed in Cache software, 2.x releases.

To specify the port number over which requests from the healing Content Engine are sent to other Content Engines in the cluster, use the **http cluster http-port** option.

URL-Based Monitoring

In the ACNS 5.2.3 software and later releases, you can configure a Content Engine to monitor the performance of specific URLs. To support this new feature, the **http monitor url url** global configuration command was added. This command enables you to specify up to 10 URLs that you want the Content Engine to monitor. The Content Engine maintains statistics about the various response characteristics for each of the monitored URLs. Use the new **show statistics http monitor** command to view these statistics.

The **http monitor url url** command has two command options, the **acceptable-delay** and **interval** options. Use the **acceptable-delay** option to specify the acceptable delay in seconds (the maximum number of seconds that the specified monitored URL should be retrieved within). The default acceptable delay is 60 seconds. The **interval** option specifies the monitoring interval (how frequently the Content Engine should monitor requests for a specific URL). The monitoring interval is specified in seconds. The default monitoring interval is 60 seconds.

**Note**

If you use the **http monitor url url** command to configure the same URL with a different interval or acceptable-delay setting, the most recently configured setting takes precedence and overrides any previously configured settings for that particular URL.

HTTP Proxy Failover

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

To designate the primary proxy, use the **primary** keyword. If several proxies are configured with the **primary** keyword, the last one configured overrides the others. Failover to a proxy server occurs in the order that the proxy servers were configured. If all the configured proxy servers fail, the Content Engine can optionally redirect requests to the origin server if the user enters the **http proxy outgoing origin-server** option. If the user has configured the **origin-server** option, the Content Engine directs HTTP requests to the original server specified in the HTTP header. If the option is not enabled, the client receives an error response. 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.

The state of the proxy servers is maintained by active monitoring, which occurs in the background. The state of the proxy servers can be seen in the CLI and syslog NOTICE messages. This interval is configured with the **http proxy outgoing monitor** option. This outgoing monitor interval is the interval of time over which the proxy servers are polled. If one of the proxy servers is unavailable, the polling mechanism waits for the connect timeout (300 milliseconds) before polling the next server.

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

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



Note

If the client is connected to the Content Engine in transparent mode and the user preserves 407 error codes by invoking the **http proxy outgoing preserve-407** command, client browsers will not recognize the 407 error codes.

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

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

The **persistent-connections** option enables persistent connections on the Content Engine. To configure the number of seconds that the Content Engine should wait for a connection response before it times out, use the **timeout** option. After you enable persistent connections, the connection does not close when no keepalive messages are sent by the Content Engine to the clients and to the edge Content Engines. Instead, the Content Engine keeps an idle connection open for the number of seconds specified by the **timeout** option. Once the transfer of data packets takes place between the Content Engine and the clients or servers, the idle period restarts. Persistent connections are not in effect unless there is a data transmission between the Content Engine and the clients or servers after the connection was established.

The **http object url-validation enable** option has a dependency with the **ip name-server** CLI command. When the **ip name-server** option is not configured (for example, during transparent proxy), the **http object url-validation enable** command is dynamically turned off. When the **ip name-server** option is configured, the **http object url-validation enable** command is turned on automatically if and only if it was enabled.

**Caution**

URL validation is on by default. We strongly recommend that you keep URL validation enabled, because disabling URL validation might make the Content Engine vulnerable to corruption from the HTTP objects in the cache.

Use the **proxy-protocols outgoing-proxy exclude** global configuration command to specify a domain for which the Content Engine should not use an upstream proxy.

The Content Engine will not use the upstream proxy for any domain that ends with the listed domain name. For example, if you specify `cisco.com`, the configured outgoing proxy server is bypassed each time that the Content Engine tries to retrieve a web page from `videos.cisco.com` or `personals.cisco.com`.

For IP addresses, enter the full IP address or use the asterisk (*) as a wildcard for IP address fields as follows:

`172.16.1.*`

`172.16.*.*`

`172.*.*.*`

The syntax `172.16.*.*` indicates that all requests to the domain host of `172.16.xxx.xxx` will be excluded. Wildcard syntax does not support 0 or the question mark (?).

The following forms of wildcard specification are not supported:

`172.*.10.2`

`172.31.1*.8`

If the client receives only a portion of the object but not the entire object because a download was prematurely halted, the client can use a Range request header to download only those missing portions of the object. The Range header specifies the byte range of the object requested. The **http smart-range enable** command enables the proxy to cache an HTTP response even if the client issues a Range request and the object is not in the cache. Use the **no** version of this command to disable this caching.

The **http smart-range max-start offset max-interval interval** byte values control these Range request conditions when this feature is enabled. The *offset* value is the maximum offset value of the first range in the client's Range request. The *interval* value is the maximum interval between any consecutive ranges in the client's request. Only if a Range request satisfies both conditions and a cache miss occurs, then the proxy issues a normal request to the origin server, caches the response (if cacheable), and sends a Range response to the client. If the response is not cacheable, then a full response is sent to the client.

Modification of Hostnames in HTTP Requests

By default, when acting as a proxy server for HTTP requests, the Content Engine may replace the original hostname in the HTTP request header with the fully qualified domain name, as defined by the **ip domain-name domain** global configuration command.

Use the **http request-header host unmodified** command to specify that the hostname parameter in the HTTP request header must *not* be converted to a fully qualified domain name. This command is a workaround for virtual servers that are not configured for use with the fully qualified hostname. Use the **no http request-header host unmodified** command to restore the default behavior.

The **show http request-header** command can be used to check if the hostname in the request header is unmodified.

HTTP TCP Keepalives

When no keepalive messages are sent by the Content Engine to the clients and to the edge Content Engines, the connection is closed. You can force the Content Engine to send keepalive probes using the **http tcp-keepalive enable** global configuration command. When the HTTP TCP keepalive

feature is enabled, the Content Engine sends TCP keepalives on idle HTTP TCP connections using keepalive configuration parameters such as a TCP keepalive timeout, a TCP keepalive probe count, and a TCP keepalive probe interval.

Examples

The following example sets the length of time that entries are valid in the authentication cache:

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

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

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

The following example specifies that the host 10.1.1.1 on port 8088 is designated as the primary proxy server and host 10.1.1.2 is designated as a backup proxy server:

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

The following example specifies that the Content Engine is configured to redirect requests directly to the origin server if all of the proxy servers fail:

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

The following example specifies that the Content Engine is configured to monitor the proxy servers every 120 seconds:

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



Note

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

The following example shows the output for the **show http proxy** command:

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

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

The following example shows the output for the **show statistics http requests** command:

```
ContentEngine# show statistics http requests
Statistics - Requests
```

	Total	% of Requests
Total Received Requests:	49103	-
Forced Reloads:	109	0.2
Client Errors:	23	0.0
Server Errors:	348	0.7
URL Blocked:	0	0.0
Sent to Outgoing Proxy:	0	0.0
Failures from Outgoing Proxy:	0	0.0
Excluded from Outgoing Proxy:	0	0.0
ICP Client Hits:	0	0.0
ICP Server Hits:	0	0.0
HTTP 0.9 Requests:	2	0.0

```

HTTP 1.0 Requests:          49101          100.0
HTTP 1.1 Requests:          0              0.0
HTTP Unknown Requests:      0              0.0
Non HTTP Requests:          0              0.0
Non HTTP Responses:         46             0.1
Chunked HTTP Responses:      0              0.0
Http Miss Due To DNS:        0              0.0
Http Deletes Due To DNS:     0              0.0
Objects cached for min ttl:  2674          5.0

```

The following example shows the output for the **show statistics http proxy outgoing** command:

```
ContentEngine# show statistics http proxy outgoing
```

```

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

```

```
Requests when all proxies were failed: 0
```

The following example shows that with the default configuration (all **cache-on-abort** thresholds disabled), client abort processing is configured to always abort downloading an object to the cache:

```
ContentEngine(config)# http cache-on-abort enable
```

The following example shows that the Content Engine is configured to always continue downloading an object to the cache (this configuration is the default):

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

The following example shows that the Content Engine is configured to use the default minimum threshold when the **cache-on-abort** option has been enabled and the threshold is set to 16 kilobytes:

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

The following example shows that the Content Engine is configured to ignore the minimum threshold:

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

The following example enables the healing mode feature by setting the HTTP port 8080 for forwarding HTTP requests to a specific port (3144) on a healing server, setting the maximum delay to wait for a response from the cluster in seconds before considering the healing request as a miss, and setting the maximum number of misses that the healing Content Engine can receive from the cluster before the healing mode is disabled at the healing client.

```

ContentEngine(config)# http cluster http-port 8080
ContentEngine(config)# http cluster heal-port 3144
ContentEngine(config)# http cluster max-delay 5
ContentEngine(config)# http cluster misses 5

```

The following example shows that the **show statistics http cluster** command displays the statistics of the healing client and the healing server. The **clear statistics http cluster** command resets the healing mode statistics.

```

ContentEngine(config)# show statistics http cluster
Healing mode max attempts      = 0
Healing mode max latency       = 10
Healing mode current cumulative misses = 0

```

Healing mode client statistics

```

-----
Client Requests Sent      = 0
Client Responses Received = 0
Client Responses Hit      = 0
Client Responses Miss     = 0
Client Responses Error    = 0
Client Responses Timeout  = 0

```

Healing mode server statistics

```

-----
Server Requests Received = 0
Server Responses Sent    = 0
Server Responses Hit     = 0
Server Responses Miss    = 0
Server Responses Error   = 0

```

```
ContentEngine(config)# clear statistics http cluster
```

The **show http cluster** command displays **max-delay**, **misses**, **http-port**, and **heal-port** values. The following example shows that the values are set to 0 and the healing client is disabled:

```

ContentEngine# show http cluster
Healing client is disabled

Timeout for responses = 10 seconds
Max number of misses allowed before stop healing mode = 0
Port number for healing request/response = 14333
Http-port to forward http request to healing server = 80

```

The following example shows that the healing client is enabled:

```

ContentEngine# show http cluster
Healing client is enabled

Timeout for responses = 10 seconds
Max number of misses allowed before stop healing mode = 999
Port number for healing request/response = 14333
Http-port to forward http request to healing server = 80

```

The following example enables caching of both basic and NTLM authenticated content on the Content Engine:

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

The following example shows how to verify which types of object caching are currently enabled on the Content Engine:

```

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

```

The following example shows how to enable NTLM object caching on a standalone Content Engine:

```
ContentEngine(config)# http cache-authenticated ntlm
```

The following example shows the **http authentication realm** command and the **show** version of the command. The **show http authentication EXEC** command can be used to view the name of the configured realm.

```

ContentEngine(config)# http authentication realm CE
ContentEngine(config)# exit
ContentEngine# show http authentication
  HTTP Authentication:

```

```

Header: 401
Realm: "CE"
Cache Timeout: 480 (minutes)
Cache Maximum entries: 4000

```

The following example shows that the Content Engine is configured to monitor the `http://www.abc.com/` URL named using the default values (an interval of 60 seconds and an acceptable delay of 60 seconds):

```
ContentEngine(config)# http monitor url http://www.abccorp.com/
```

The following example shows that the Content Engine is configured to monitor the URL named `http://www.abc.com/`. The Content Engine is configured to wait up to 100 seconds for the URL to be retrieved and to monitor requests for this URL every 100 seconds.

```
ContentEngine(config)# http monitor url http://www.abc.com/ acceptable-delay 100
interval 100
```

If it takes more than 100 seconds for the URL to be retrieved, the specified acceptable delay is exceeded. The Content Engine tracks the response time (minimum and maximum delay time) and the number of times that the acceptable delay is exceeded for a particular URL. These statistics are shown in the output from the **show statistics http monitor EXEC** command.

The following example shows how to configure the host for the LDAP server daemon:

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

The following example shows how to configure the host for the RADIUS server:

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

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

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

The following example shows how to disable the DNS cache with the **no** form of the **http dns-cache** command:

```
ContentEngine(config)# no http dns-cache size
```

The following example shows how to exclude a domain from an outgoing proxy:

```
ContentEngine(config)# proxy-protocols outgoing-proxy exclude cisco.com
```

The following examples show the **http request-header host unmodified** and **show http request-header** commands:

```

ContentEngine(config)# http request-header host unmodified
ContentEngine(config)# exit
ContentEngine# show http request-header
HTTP request header
-----
Host:          Don't append a default domain name (cisco.com) to hostname
                string if it is not a fully qualified domain name
ContentEngine(config)# no http request-header host unmodified
ContentEngine(config)# exit
ContentEngine# show http request-header
HTTP request header
-----
Host:          Append a default domain name (cisco.com) to hostname
                string if it is not a fully qualified domain name

```


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

```
ContentEngine(config)# no http authenticate-strip-ntlm
ContentEngine(config)# http cache-authenticated ntlm
ContentEngine# show http cache-authenticated ntlm
Basic authenticated objects are not cached.
NTLM authenticated objects are cached.
21
```

Related Commands

- acquirer (EXEC mode)**
- bypass gateway**
- clear dns-cache**
- dnslookup**
- ip name-server**
- proxy-protocols**
- rule no-proxy**
- rule use-proxy**
- show acquirer**
- show http**
- show http methods**
- show http proxy**
- show statistics http dns-cache**
- show statistics http monitor**
- show statistics http proxy outgoing**
- show statistics http requests**

http custom-error-page

To configure HTTP custom error messages, use the **http EXEC** command.

http custom-error-page { **download** *message url* | **reset** { **all** | *message* } | **upload** { *ip-address* | *hostname* } *dirname filename message* }

Syntax Description

download	Copies the custom error message file to the Content Engine from the specified URL. To change the text for a specific message, use this option to identify the message 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 page for the specified message.
<i>message</i>	Type of custom error message (for example, ftp-put-error). See Table 2-5 for a list of these custom error messages.
<i>url</i>	Source of the custom error file. The file size cannot exceed 16 KB.
reset	Reverts to the default error page.
all	Reverts all custom error pages to default error pages.
<i>message</i>	Message that you want to revert to the default page.
upload	Uploads the custom error message file to the specified host, directory, and file.
<i>ip-address</i>	IP address of the host to which to copy the error page.
<i>hostname</i>	Hostname to which to copy the error page. The host should be reachable and allow copying a file to the specified directory.
<i>dirname</i>	Directory name to which to copy the error page.
<i>filename</i>	Filename to which to copy the error page.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

Use this command to work with HTTP custom error messages. Custom error messages let you display text messages of your choice in response to various system events.

To change the text for a specific message, use the **http custom-error-page download** command, 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 page for the specified message.

To reset a specific message to the default message text, use the **http custom-error-page reset** command and identify the message that you want to revert to the default page.

To copy the current contents of a specified message page, use the **http custom-error-page upload** command and identify the hostname or IP address, the directory, and the filename where you want to copy the specified message. The host should be reachable and allow copying a file to the specified directory.

The messages and their meanings are shown in [Table 2-5](#).

Table 2-5 Custom Error Page Messages

Message Identifier	Usage
blocked-dueto-filter-error	Error response when a request is blocked because of a filter.
cache-read-error	Error response when a cache file system (cfs) read fails.
cache-write-error	Error response when a cfs write fails.
cdn-not-found-error	Error response when the ACNS network is not found.
client-access-denied-msg	Error response when client access is denied.
client-connection-broken-error	Error response when a client connection is lost.
cr-domain-not-found-err	Error response when the Content Router could not be found.
cr-general-error	Error response when the Content Router is not operational.
cr-not-in-cz-error	Error response when the Content Router is not found in the coverage zone.
cr-unavailable-error	Error response when the Content Router is not available.
dns-not-available-error	Error response when DNS is unavailable for resolution.
error-signature	Signature message that is appended to the final error page that is displayed to the end user (added in the ACNS 5.3 software release). If you do not create a customized error-signature message, then the default error-signature message is used. For more information about creating a customized error-signature message, see the “Creating a Custom Error-Signature Message for HTTP Custom Error Pages” section on page 2-228.
expect-failed-error	Error response when the Expect specifier in the HTTP request header cannot be met.
ftp-bad-login-error	Error response when an FTP login fails.
ftp-bad-url-error	Error response when an FTP request receives a bad URL.
ftp-disabled-error	Error response when FTP is disabled.
ftp-failure-error	Error response when FTP fails.
ftp-internal-error	Error response when an FTP interval is exceeded.
ftp-not-found-error	Error response when an FTP file is not found.
ftp-put-created-msg	Error response when an FTP PUT is successful.
ftp-put-error	Error response when an FTP PUT fails.
ftp-put-modified-msg	Response when an FTP update is successful.
ftp-unavailable-msg	Error response when an FTP file is unavailable.
http-blocked-port-msg	Error response when an HTTP request comes through a blocked port.

Table 2-5 Custom Error Page Messages (continued)

Message Identifier	Usage
https-blocked-port-msg	Error response when an HTTPS request comes through a blocked port.
icap-processing-error	Error response when an error occurred in ICAP processing.
invalid-port-error	Error response when an invalid port is accessed.
looped-req-error	Error response when a looped request is unsuccessful.
not-enough-resources-error	Error response when enough resources are not available for the request process.
not-in-cache	Error response when the object is not found in the cache.
offline-miss-error	Error response when an offline Content Engine finds a cache miss.
outgoing-proxy-fail-error	Error response when all outgoing proxies fail.
proxy-allow-domain-error	Error response when the domain is not allowed to authenticate in proxy mode (added in the ACNS 5.3 software release).
proxy-no-default-domain-error	Error response when there is no default domain available to authenticate in proxy mode (added in the ACNS 5.3 software release).
proxy-unauthenticated-error	Error response when proxy authentication fails.
radius-redirect-error	Error response for a RADIUS redirect message.
request-blocked-msg	Error response when a request is blocked.
request-malformed-error	Error response when request headers are malformed.
rev-dns-not-available-msg	Error response when DNS is not available.
server-connection-broken-error	Error response when the server connection is lost.
ssl-server-error	Error response when the SSL handshake with the server fails.
unsupported-cr-method-error	Error response when an unsupported Content Router method is used.
www-allow-domain-error	Error response when the domain is not allowed to authenticate (added in the ACNS 5.3 software release).
www-no-default-domain-error	Error response when there is no default domain available to authenticate (added in the ACNS 5.3 software release).
www-unauthenticated-error	Error response when server authentication fails.

Creating a Custom Error-Signature Message for HTTP Custom Error Pages

In the ACNS 5.3 software release, the ability to include an error signature on the HTTP custom error pages was added. To support this feature, the **error-signature** message identifier was added to the list of custom error messages (listed in [Table 2-5](#)).

The following is the default error signature that will be appended to the final error message page sent to the end user:

```
<br clear="all">
<hr noshade size=1>
Generated %t by
(<a href="http://www.cisco.com/">Application and Content Networking System Software
5.3.0</a>)
</BODY></HTML>
```

You can create a custom error signature that will be appended to the final error message page instead of the default error signature. The following example shows an example of a custom error signature:

```
The request %R , from the Client %i does not conform to the HTTP Request.
Please refer RFC-XXX, for further information on request format.
</body>
</html>
```

After creating a custom error signature, you use the **http custom-error-page download error-signature EXEC** command to copy the file that contains the custom error signature to the Content Engine as follows:

```
ContentEngine# http custom-error-page download error-signature ftp://somewhere/sig.htm
```

A custom error signature message supports dynamic data. This dynamic data is specific to the particular request and response that is to be shown with the error messages.

[Table 2-6](#) lists the argument specifiers for the error signature. These argument specifiers are replaced with the corresponding value when the error signature is appended to the final error message page that is sent to the end user.

Table 2-6 *Argument Specifiers for Error Signatures*

Argument Specifier	Description
c	Squid error code
d	Seconds elapsed since the request was received
e	Error number
E	Error string
h	Cache hostname (hostname of the Content Engine)
H	Server hostname
i	Client IP address
I	Server IP address
M	Request method
p	URL port # (number)
P	Protocol
R	Full HTTP request
t	Local time
T	Coordinated Universal Time (UTC)
U	URL without password

The following example shows how the template will be transformed into the final error message page that is sent to the end user. You create a custom error signature that uses the R and i argument specifiers. The template contains the following for a request malformed error:

```
<h1> Malformed request </h1>
The request %R , from the Client %i does not conform to the HTTP Request.
Please refer RFC-XXX, for further information on request format.
</body>
</html>
```

When a request malformed error occurs for the client with the IP address of 192.168.192.161, the following custom error signature is appended to the final error message page that is sent to the end user (client 192.168.192.161):

```
<h1> Malformed request </h1>
The request GET http://www.abccorp.com HTTP/1.0, from the client 192.168.192.161
does not conform to the HTTP Request.
Please refer RFC-XXX, for further information on request format.
</body>
</html>
```

Because the R and i argument specifiers were specified as part of the custom error signature, the actual full HTTP request (http://www.abccorp.com HTTP/1.0) and the IP address of the client (192.168.192.161) are included in the error signature of the final error message page that is sent to the end user.

Examples

The following example copies a custom error page to the Content Engine for the cache-read-error page:

```
ContentEngine# http custom-error-page download
http://www.myserver.com/errors/cache-read-error.txt cache-read-error
ContentEngine#
```

The following example copies the current contents of the cache-read-error page to a file in the *errors* directory on the host with the IP address 192.168.1.1:

```
ContentEngine# http custom-error-page upload 192.168.1.1 /errors cache-read-error.txt
cache-read-error
ContentEngine#
```

The following example resets the cache-read-error page to the default text:

```
ContentEngine# http custom-error-page reset cache-read-error
```

When the authentication mechanism is NTLM, the Content Engine sends the proxy-allow-domain-error/www-allow-domain-error error message to the end user if the user-supplied domain does not match the configured domains. You can customize the error that is sent to the end user as follows:

```
ContentEngine# http custom-error-page download proxy-allow-domain-error
http://myserver.com/errors/proxy-allow-domain.txt
WORD URL from where the file will be fetched, File size cannot exceed 16K
```

With certain browsers (for example, with the Netscape browser), the end user must specify the domain if the Content Engine does not have an NTLM default domain configured. If there is no default domain available for authentication, the proxy-no-default-domain-error/www-no-default-domain-error message is sent to the end user. You can customize the error message that is sent to the end user as follows:

```
ContentEngine# http custom-error-page download proxy-no-default-domain-error
http://myserver.com/errors/proxy-no-default-domain-error.txt
WORD URL from where the file will be fetched, File size cannot exceed 16K
```

```
ContentEngine# http custom-error-page download www-no-default-domain-error
http://myserver.com/errors/www-no-default-domain-error.txt
WORD URL from where the file will be fetched, File size cannot exceed 16K
```

The following example resets all of the configured custom error pages to the default text:

```
ContentEngine# http custom-error-page reset all
```

Related Commands show http custom error-page

https (EXEC)

To create, remove, and import certificates and private keys when using the Content Engine as an HTTPS server, use the **https** EXEC command.

```
https {cert cert-name {create | import url URL | remove} | certgroup certgroup-name {add-cert | create | remove} | key key_name {create | import url URL | remove}}
```

Syntax Description

cert	Enables creating, removing, and importing certificates.
<i>cert-name</i>	Name of the certificate object.
	Note A certificate object is a container that stores the certificate. You create an empty certificate object when you do not specify the URL from which the certificate must be imported.
create	Creates a certificate object with the name specified.
import	Imports a certificate from an external source.
url	Enables the use of a URL to point to the location of the private key.
<i>URL</i>	URL (HTTP, FTP-over-HTTP, or HTTPS) that points to the location of the private key.
remove	Removes a certificate object with a given name.
certgroup	Enables adding, creating, or removing a certificate group.
<i>certgroup-name</i>	Name of the certificate group.
add-cert	Adds a certificate to the certificate group, starting from the end entity's certificate to the root certificate authority's (CA) certificate.
create	Creates a certificate group with the specified name.
remove	Removes a certificate group with the specified name.
key	Enables creating, removing, and importing a private key.
<i>key_name</i>	Name of the private key object.
create	Creates a private key object with the given name.
import	Imports a private key from an external source into the key object.
	Note A key object is a container that stores the private key. You create an empty key object when you do not specify the URL from which the HTTPS key must be imported.
url	Enables the use of a URL to point to the location of the private key.
<i>URL</i>	URL (HTTP, FTP-over-HTTP, or HTTPS) that points to the location of the private key.
remove	Removes a key object with the given name.

Defaults

No default behavior or values

Command Modes

EXEC (privileged)

Usage Guidelines

In the ACNS software, you can configure your Content Engine to use a set of Secure Sockets Layer (SSL) certificates and keys to enable the Content Engine to act as an origin HTTPS server. This method can reduce WAN traffic and increase data security, because authorized clients from remote branch offices can use HTTPS caching solutions deployed on the Content Engine at their branch offices. HTTPS proxy mode enables the Content Engine to service HTTPS requests sent by the web clients, which have been configured to use an HTTPS proxy server.

The Content Engine decodes HTTPS traffic from a client and performs normal HTTP operations on it, such as caching and request processing. The Content Engine initiates HTTPS connections to an origin server and fetches the content from origin servers when a cache miss occurs.

For specific requested content to be cached, you must import the proper certificates and keys for these sites (HTTPS servers) into the Content Engine and instruct the Content Engine to cache these sites. The Content Engine presents the certificate to HTTPS clients that make requests to the HTTPS server.

**Note**

For information on the restrictions concerning the use of certificates, keys, and certificate groups, see Chapter 8 of the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments*.

A digital certificate is a credential that allows the Content Engine to be presented to an HTTPS client as the original HTTPS server.

You can assign a certificate and associate a key with the HTTPS server if you have configured the Content Engine with the **https server** global configuration command. The Content Engine presents the certificate to HTTPS clients that make requests to the HTTPS server.

The Content Engine accepts certificates in Privacy-Enhanced Mail (PEM) format which is used by Apache servers, and Public-Key Cryptography Standards (PKCS) #12 format, which is used by Microsoft Internet Information Services (IIS).

The Content Engine uses PEM format internally and automatically converts certificates in PKCS #12 format to PEM format. If you need to use a certificate in a different format, first convert it to one of these supported formats.

Use the **https cert EXEC** command to create certificate objects with a given name, to import a certificate from external sources into a certificate object, or to remove existing certificate objects. Use any name you like for the certificate and the key. Use a URL where the Content Engine can obtain the certificate and key files.

A certificate object is a container that stores the certificate. You create an empty certificate object when you do not specify the URL from which the certificate must be imported. Certificate object names can consist of Arabic numerals, uppercase and lowercase letters, underscores, and hyphens. Certificate object names must begin with a letter, a numeral, an underscore (_), or a hyphen (-), and have a limit of 64 characters. You need to create a certificate object before you can import the certificate from an external source and associate it with an HTTPS server. Only certificate names are stored in the CMS database. Actual certificates are stored only on the Content Engine.

**Note**

Once you create a certificate object, you cannot modify the name. To modify a name, you must delete the existing certificate and create a new one.

**Note**

It is possible for a certificate object to exist without its certificate being imported. However, only certificate objects with imported certificates can be associated with an HTTPS server or added to a certificate group. Two different certificates can be imported from the same URL and two different HTTPS servers can be associated with the same certificate.

If the external location from which the certificate is to be imported into the certificate object is password protected, you must specify the username required to access the external source from which the certificate is being imported and the password used to authenticate users who want to gain access to the external source from which the certificate is being imported. When the external source is password protected, the URL should use any one of the following formats:

- `ftp://user:password@domainname/path`
- `http://user:password@domainname/path`
- `https://user:password@domainname/path`

The **https certgroup** EXEC command allows you to create or remove certificate groups or import a certificate from an external source and add it to an existing certificate chain. Certificate groups constitute a chain of trust relationships from the root Certificate Authority to the end entity. Each one of the certificates in a certificate group, except the end entity's certificate, signs and trusts the next one in the chain. An end entity's certificate can be trusted only if all certificates in the certificate group leading to this certificate can be trusted. A certificate group can be configured on the Content Engine (acting as an HTTPS server) to support HTTPS caching just like a single certificate, but with the added benefit that the client does not need to have all certificates locally. A certificate group can also be used to verify and authenticate an HTTPS server by comparing the server's certificates to those certificates in the certificate group.

Use the **https certgroup** *certgroup-name* **create** EXEC command to create a certificate group with a given name. Certificate group names can consist of Arabic numerals, uppercase and lowercase letters, underscores, and hyphens. Certificate group names must begin with a letter, a numeral, an underscore (_), or a hyphen (-), and have a limit of 64 characters.

**Note**

Once you create a certificate group, you cannot modify the name. To modify a name, you must delete the existing certificate group and create a new one.

**Note**

Two different certificate groups can have the same combination of HTTPS certificates.

Use the **https key** EXEC command to create a private key object with a given name, to import a private key object from an external source, or to remove a private key object with a given name. A private key is the secret half of a key pair used in a public key encryption algorithm. Private keys are typically used to encrypt a symmetric session key, digitally sign a message, or decrypt a message that has been encrypted with the corresponding public key. PKCS # 12 defines a portable format for storing or transporting a user's private keys and certificate information. The private key that the Content Engine uses to act as an origin HTTPS server must match the selected authentication certificate.

A key object is a container that stores the private key. You create an empty key object when you do not specify the URL from which the HTTPS key must be imported. Key object names can consist of Arabic numerals, uppercase and lowercase letters, underscores, and hyphens. Key object names must begin with a letter, a numeral, an underscore (_), or a hyphen (-), and have a limit of 64 characters.

You need to create a private key object before you can import the HTTPS key from an external source and associate it with an HTTPS server. Only HTTPS key names are stored in the CMS database. Actual certificates are stored only on the Content Engine.

**Note**

Once you create a key object, you cannot modify the name. To modify a name, you must delete the existing key and create a new one.

**Note**

It is possible for a key object to exist without its key being imported. However, only key objects with imported keys can be associated with an HTTPS server. Two different keys can be imported from the same URL and two different HTTPS servers can be associated with the same key.

If the external location from which the private key is to be imported into the key object is password protected, you must specify the username required to access the external source from which the key is being imported and the password used to authenticate users who want to gain access to the external source from which the key is being imported. When the external source is password protected, the URL should use any one of the following formats:

- `ftp://user:password@domainname/path`
- `http://user:password@domainname/path`
- `https://user:password@domainname/path`

Examples

The following example shows how to load the certificates and private key on the Content Engine:

```
ContentEngine# https cert mycert create
ContentEngine# https cert mycert import http://www.myca.com/myservercert
ContentEngine# https cert mykey create
ContentEngine# https cert mykey import http://www.myca.com/myprivatekey
```

Related Commands

`show https`

https (global configuration)

To configure the Content Engine for HTTPS proxy services, use the **https** global configuration command. To disable individual options, use the **no** form of this command.

https destination-port { **allow** { *ports* | **all** } | **deny** { *ports* | **all** } }

https proxy { **incoming** *ports* | **outgoing** { **connection-timeout** *timeout* | **host** { *hostname* | *ip-address* } *port* [**primary**] | **monitor** *interval* | **origin-server** } }

https tcp-rw-timeout *timeout*

no https { **destination-port** { **allow** { *ports* | **all** } | **deny** { *ports* | **all** } } | **proxy** { **incoming** *ports* | **outgoing** *host* { **connection-timeout** | **host** { *hostname* | *ip-address* } *port* [**primary**] | **monitor** | **origin-server** } } | **tcp-rw-timeout** }

Syntax Description

destination-port	Specifies the destination port restrictions.
allow	Allows HTTPS traffic to specified ports. Ports 443 and 563 are allowed by default.
<i>ports</i>	Port numbers (1–65535). You can configure a maximum of eight ports.
all	Specifies all ports.
deny	Denies HTTPS traffic to specified ports. Port numbers below port 1024 are denied by default.
proxy	Sets configuration parameters for proxy mode.
incoming	Sets the configuration for incoming proxy-mode requests.
<i>ports</i>	Port numbers (1–65535) to listen for HTTPS requests. Up to eight ports can be configured.
outgoing	Sets the configuration to direct outgoing requests to another proxy server.
connection-timeout	Specifies the timeout value (in microseconds) used for probing outgoing proxy servers.
<i>timeout</i>	Timeout value (in microseconds) used for probing outgoing proxy servers (200–5000000).
host	Uses the outgoing HTTPS proxy. You can configure a maximum of eight outgoing proxy servers.
<i>hostname</i>	Hostname of the outgoing proxy.
<i>ip-address</i>	IP address of the outgoing proxy.
<i>port</i>	Port number of the outgoing proxy (1–65535).
primary	Sets the specified host as the primary outgoing HTTPS proxy server. If several servers (hosts) are configured with the primary keyword, the last one configured becomes the primary outgoing HTTPS proxy server for the Content Engine.
monitor	Specifies the interval at which the outgoing proxy servers are to be monitored.
<i>interval</i>	Monitoring interval in seconds (10–300).
origin-server	Specifies that the origin server must be used if all outgoing proxy servers fail.

tcp-rw-timeout	Forces the Content Engine to send HTTPS TCP keepalive messages during the time specified with the <i>timeout</i> parameter.
<i>timeout</i>	Maximum time period (in seconds) during which the Content Engine sends HTTPS TCP keepalive messages. (1–3600).

Defaults

https destination-port allow *ports*: 443 and 563

https destination-port deny *ports*: port numbers below port 1024

https tcp-rw-timeout *timeout*: 5 minutes

Command Modes

global configuration

Usage Guidelines

The HTTPS protocol is essentially the HTTP protocol running over a Secure Sockets Layer (SSL) transport. SSL provides a secure channel between two devices (for example, a client and a server). SSL uses public-key cryptography to ensure the security and privacy of the exchange between the client browser and the server. HTTPS uses a unique URL that begins with `https://` (for example, `https://abc.com`). The default port number for HTTPS is port 443 instead of port 80, which is the default port for HTTP.

The ACNS software supports HTTPS in the following two scenarios:

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

Content Engines can either SSL terminate or tunnel HTTPS client requests to the origin HTTPS server, as described in the following sections.

SSL Termination of HTTPS Client Requests

If the Content Engine SSL terminates an HTTPS client request, the Content Engine decrypts the SSL-encrypted data. By decrypting the SSL-encrypted data, the Content Engine can see the HTTPS client request in plain text, which allows the Content Engine to perform numerous HTTP processing tasks (for example, caching, rule processing, and filtering) on such requests. Content Engines running the ACNS 5.1 software and later releases, support this SSL-termination feature. Content Engines running the ACNS 5.2 software and later releases, can also rewrite and redirect HTTPS requests as defined by the specified rules. If the Content Engine terminates an HTTPS request, it can apply most of the rules on the HTTPS request. (The **no-proxy**, **use-icap-service**, and **use-proxy** rule actions are not supported for HTTPS caching.)

For the SSL-termination feature to work properly, you must install the SSL certificate and private key of the origin HTTPS servers on the Content Engine. The SSL-termination feature works in both transparent mode and proxy mode if the Content Engine has the correct certificates and private keys installed. For specific requested content to be cached, you must import the proper certificates and keys for these origin HTTPS servers into the Content Engine and configure the Content Engine to cache the content from these origin HTTPS servers. For Content Engines, this process is performed through the Content Engine CLI, as described in the [“https \(EXEC\)” section on page 2-232](#).

In the ACNS 5.2 software and later releases, the Content Engine SSL terminates HTTPS requests in WCCP mode and in manual proxy mode if the requested HTTPS servers are configured on the Content Engine and tunnels the rest of the HTTPS traffic.

Tunneling of HTTPS Client Requests

The ACNS software supports tunneling for HTTPS client requests. In this mode, the Content Engine can only support limited processing on the HTTPS client requests (for example, no caching support and only limited filtering support).

The ACNS 3.0 software and later releases support HTTPS tunneling through the CONNECT method, which is the standard HTTPS tunneling method that is defined in the HTTP specification.

The ACNS 5.1.5 software and later releases support tunneling of transparently redirected native HTTPS traffic. The Content Engine can accept native HTTPS traffic from the client and tunnel such requests to the origin HTTPS server. Even though the Content Engine can apply filtering on such traffic, it does not see the actual HTTPS content because the certificate and private key of the origin HTTPS server is not installed on the Content Engine.

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

Table 2-7 shows CLI commands associated with HTTPS proxy features. The order in which the CLI commands are entered is not important.

Table 2-7 **HTTPS Proxy Features**

HTTPS Proxy Feature	Related CLI Command (Abbreviated Syntax)
Supports up to eight incoming proxy ports.	https proxy incoming ports <i>1–65535, ports, . . .</i>
Configures a WCCP service and an HTTPS incoming proxy on the same port. Shares the proxy port with transparent services.	https proxy incoming ports <i>1–65535</i> wccp custom-web-cache . . .
Denies unwanted access to any destination HTTPS port.	no https destination-port allow 443 563 https destination-port deny all
Configures the outgoing HTTPS proxy server, using the global exclude option for HTTPS proxy.	proxy-protocols outgoing-proxy exclude list word https proxy outgoing host { <i>hostname</i> <i>ip_address</i> } <i>port 1–65535</i>
Uses the default outgoing HTTPS proxy, if available.	proxy-protocols transparent default-server
Uses the outgoing HTTPS proxy server from an original request.	proxy-protocols transparent original-proxy
Returns the incoming HTTPS request to the sending client during a cache miss.	proxy-protocols transparent reset

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

When the Content Engine is configured to use an HTTPS outgoing proxy with the **https proxy outgoing host** command, all incoming HTTPS requests are directed to this outgoing proxy. In the ACNS 5.1.x software and earlier releases, you could only configure the Content Engine to use one outgoing HTTPS proxy server. In the ACNS 5.2 software, the ability to configure the Content Engine to use up to eight outgoing HTTPS proxy servers was added.

**Note**

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

The **proxy-protocols outgoing-proxy exclude** command specifies a global proxy exclude domain effective for all proxy server protocols including HTTPS.

The Content Engine applies the following logic when an outgoing proxy server is configured:

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

When a Content Engine intercepts a proxy request intended for another proxy server and there is no outgoing proxy configured for HTTPS, and the **proxy-protocols transparent default-server** command is invoked, the Content Engine addresses the request to the destination server directly and not to the client's intended proxy server. However, if the **proxy-protocols transparent reset** command is configured on the Content Engine and a cache miss occurs, all transparently intercepted requests sent by clients are returned to the client and requested objects are not delivered.

Content Engine Security

To prevent unwanted access to any destination HTTPS port when a request is going through the Content Engine, use the following command sequence:

no https destination-port allow 443 563

https destination-port deny all

This command sequence denies access to any port above and below 1024. Ports 443 and 563 (the standard HTTPS ports) must be explicitly denied access using the **no https destination-port allow 443 563** command.

**Note**

TCP and UDP packets use port numbers defined by the application in use. Typically, the port range 0–255 is used for standard public applications such as FTP, and the port range 256–1023 is used by companies for nonstandard applications. For instance, FTP uses port 21, and Telnet uses port 23. Port numbers from 1024 through 65,536 are unregulated, so you should configure the **https destination-port deny ports** command to specifically deny access through any port number.

For example, when these commands are configured on the Content Engine and the request to access port xxx at <http://banking.wellsfargo.com> is redirected to this Content Engine, the connection to port xxx is denied. This configuration is valid either in the transparent deployment scenario, in which requests are redirected to the Content Engine, or in HTTPS proxy server mode, when the user makes the requests directly to the Content Engine.

HTTPS Proxy Failover

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

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

If the primary outgoing proxy server fails to respond to the HTTPS 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 HTTPS requests to the origin server specified in the HTTP header if you have used the **https 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. Proxy servers cannot be used simultaneously. You can view the state of the outgoing HTTPS proxy servers in syslog NOTICE messages and with the **show https 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 **https 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 HTTPS cache miss, the following occurs:

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



Note

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

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

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 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 **https proxy outgoing monitor** command to specify how frequently the Content Engine polls the specified outgoing HTTPS proxy servers.

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

Statistics Reporting

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

Transaction Logging

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

HTTPS TCP Keepalives

When no keepalive messages are sent by the Content Engine to the clients and to the edge Content Engines, the connection is closed. You can force the Content Engine to send keepalive probes using the **https tcp-rw-timeout** global configuration command. When the HTTPS TCP keepalive feature is enabled, the Content Engine sends TCP keepalives on idle HTTPS TCP connections using keepalive configuration parameters such as the TCP keepalive timeout, the TCP keepalive probe count, and the TCP keepalive probe interval.

The **https tcp-rw-timeout** *timeout* command allows you to configure a maximum read/write timeout of 3600 seconds; HTTPS keepalives are sent for the specified period. For HTTPS connections, the default timeout value is 5 minutes.

Examples

The following example shows that the Content Engine is configured as an HTTPS proxy server and accepts HTTPS requests on port 8081. Only a single port is supported in the HTTPS protocol.

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

The following example shows that the Content Engine is configured to forward HTTPS requests to an outgoing proxy server (10.1.1.1) on port 8880:

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

The following example shows that a domain name is excluded from being forwarded to an outgoing proxy server:

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

The following example shows that the **show https all** command is used to display all HTTPS-related information residing on your local Content Engine:

```
ContentEngine# show https all
Incoming HTTPS proxy:
Incoming Proxy-Mode:
    Not servicing incoming proxy mode connections.
Outgoing HTTPS proxy:
    Not using outgoing proxy mode.
Destination port restrictions:
Allow    443    563
HTTPS caching certificate information:
HTTPS caching certificate group information:
HTTPS caching private key information:
Display all https server caching information:
ContentEngine#
```

The following example shows that Content Engine A is configured to send its missed HTTPS traffic (cache misses for browser requests for HTTPS content [HTTPS-over-HTTP requests]) to the host 10.1.1.1 on port 8088. Host 10.1.1.1 is explicitly designated as the primary outgoing HTTPS proxy server for Content Engine A. Host 10.1.1.2 is configured as a backup outgoing HTTPS proxy server for Content Engine A.

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

The following example shows how to configure the Content Engine to redirect HTTPS requests directly to the origin server if all of the proxy servers fail:

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

The following example shows how to configure the Content Engine to monitor the outgoing HTTPS proxy servers every 120 seconds:

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

Related Commands

```
http proxy
proxy-protocols
show http proxy
show proxy-protocols
```

https server

To configure the Content Engine to act as an origin HTTPS server, use the **https server** global configuration command. This command enables HTTPS configuration mode. To disable individual options, use the **no** form of this command.

```
https server name {cert cert-name | certgroup {chain | serverauth} certgroup-name | enable |
host {hostname | ip-address} | key {keyname | password password } | protocol-version
{ssl2-only | ssl23-tls1 | ssl3-only | tls1-only} | serverauth {enable | ignore
{cert-not-yet-valid | domain-name | expired-date | invalid-ca}} | session-cache {size size |
timeout timeout}}
```

```
no https server name {cert cert-name | certgroup {chain | serverauth} certgroup-name | enable |
host {hostname | ip-address} | key {keyname | password password} | protocol-version
{ssl2-only | ssl23-tls1 | ssl3-only | tls1-only} | serverauth {enable | ignore
{cert-not-yet-valid | domain-name | expired-date | invalid-ca } | session-cache {size size |
timeout timeout}}
```

Syntax Description

<i>name</i>	Name of the HTTPS server.
cert	Sets the certificate for the HTTPS server.
<i>cert-name</i>	Certificate name.
certgroup	Sets the certificate chains and authentication needed to access the HTTPS server.
chain	Sets the certificate chain needed to access the HTTPS server.
serverauth	Sets the certificate chains and authentication needed to access the HTTPS server.
<i>certgroup-name</i>	Certificate chain name.
enable	Enables caching and SSL termination for the HTTPS server.
host	Configures the hostname for the HTTPS server.
<i>hostname</i>	Hostname of the HTTPS server.
<i>ip-address</i>	IP address of the HTTPS server.
key	Configures the private key for the HTTPS server.
<i>keyname</i>	Private key name.
password	Sets the password needed to decrypt the private key file.
<i>password</i>	Password needed to decrypt the private key file.
protocol-version	Sets the protocol versions for communication between clients and HTTPS servers.
ssl2-only	Sets the use of the Secure Sockets Layer (SSL) Version 2 protocol only.
ssl23-tls1	Sets the use and understanding of SSL Version 2 or Version 3 and Transport Layer Security (TLS) Version 1 protocol.
ssl3-only	Sets the use and understanding of the SSL Version 3 protocol only.
tls1-only	Sets the use and understanding of the TLS Version 1 protocol only.
serverauth	Configures HTTP server authentication commands.
enable	Enables the HTTPS server authentication.
ignore	Ignores particular errors in the HTTPS server authentication.

cert-not-yet-valid	Ignores errors caused by using the certificate before it is valid.
domain-name	Ignores errors due to domain name mismatch.
expired-date	Ignores certificate expiration errors.
invalid-ca	Ignore errors caused by an unrecognized Certificate Authority (CA).
session-cache	Configures SSL session caching parameters.
size	Sets the number of SSL caching entries.
<i>size</i>	Number of SSL session caching entries (0–20000).
timeout	Sets the SSL session cache timeout.
<i>timeout</i>	SSL session cache timeout in seconds (0–86400).

Defaults

The HTTPS server is disabled by default. When the HTTPS server is enabled, the following default values apply:

protocol version: sslv23-tlsv1

serverauth: enabled

serverauth ignore: none

session cache size: 10000

session cache timeout: 3600

Command Modes

global configuration

Usage Guidelines

The **https server *name*** command invokes a configuration mode that you use to enter commands for configuring a specific instance of the HTTPS server. After entering the HTPTPS configuration mode, you can enter the specific commands required to configure the HTTPS server without having to repeatedly type the **https server *name*** command.

Use the **https server *name*** command to configure an HTTPS server and configure a Content Engine to act as an origin HTTPS server. This configuration can reduce the WAN traffic and increase data security, because authorized clients from remote branch offices can use HTTPS to access their own Content Engines configured as HTTPS servers, which reside in central locations.

The Content Engine will decode HTTPS traffic from a client and perform normal HTTP operations, such as caching and request processing. The Content Engine initiates HTTPS connections to an origin server and fetches the content from origin servers upon cache miss or cache validation.

The **https server *name* cert** command configures a Content Engine to use a specific SSL certificate so that the Content Engine can act as an origin HTTPS server.

The **https server *name* certgroup chain** command identifies the certificate group for the HTTPS server. The certificate group is included in the SSL handshake that is sent to SSL clients. Only one certificate or certificate group can be selected for an HTTPS server at any time.

The **https server *name* enable** command enables HTTPS server caching. Using the **no** form of this command disables caching if it is enabled.

The **https server *name* host** command specifies the IP address or fully qualified domain name for the origin HTTPS server.

The **https server name key** command specifies the private key to use so that the Content Engine can act as an origin HTTPS server. This private key must match the selected certificate. If the key file is encrypted, use the **password** option to specify the password required to decrypt it.

The **https server name protocol-version** command specifies the SSL protocol version used to control communication between the SSL clients and the HTTPS server.

The **https server name serverauth enable** command causes verification of the origin HTTPS server's certificate, which is the default behavior.

You can use other options provided by the **serverauth** keyword to ignore authentication errors such as invalid certification, domain name mismatches, certificate expiration errors, and unrecognized Certificate Authorities (CAs).

**Note**

For information on how to enable an HTTPS on a Content Engine in an ACNS network see the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

Troubleshooting Transparent HTTPS Caching

Use the **show statistics https requests EXEC** command to determine how many HTTPS requests go through the Content Engine.

Use the **show bypass list EXEC** command to determine whether the origin HTTPS server has been added to the WCCP accept list.

Problems with certificate verification between the HTTPS client and the HTTPS server running on the Content Engine will cause the user to see a prompt asking if the certificate should be trusted.

Problems with certificate verification between the Content Engine and the origin HTTPS server will prevent the connection because no user is available to accept the certificate.

Use the **show statistics https error EXEC** command to determine if problems have occurred during the certificate verification between the Content Engine and the origin HTTPS server.

Use the **serverauth ignore HTTPS** configuration mode command to disable specific verification errors.

Use the option (cert-not-yet-valid, domain-name, expired-date, or invalid-ca) that identifies the type of errors that you want to ignore.

Use the option that identifies the type of errors that you want to ignore.

To completely disable the server-side certificate verification, enter the following command in HTTPS configuration mode:

**Caution**

Use caution when disabling the certificate verification, because connecting to an HTTPS server without verifying its certificate may create a security risk.

**Note**

For information on how transparent HTTPS works, see Chapter 7 of the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments*.

**Note**

If specific requested content is to be cached, you must import the proper certificates and keys for these sites onto the Content Engine and instruct the Content Engine to cache these sites.

Examples

The following example shows how to load the certificates and private key from EXEC mode on the Content Engine:

```
ContentEngine# https cert mycert create
ContentEngine# https cert mycert import http://www.myca.com/myservercert
ContentEngine# https cert mykey create
ContentEngine# https cert mykey import http://www.myca.com/myprivatekey
```

**Note**

You can use any name for the certificate and the key. Replace the URLs in the example with the URLs where the Content Engine can obtain the certificate and key files.

The Content Engine uses PEM format internally and automatically converts certificates in PKCS #12 format to PEM format. If you need to use a certificate in a different format, first convert it to one of these supported formats.

The following example shows that when you enter the **https server test** command from the global configuration prompt, the CLI prompt changes:

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

The following example shows that you can then enter the specific commands required to configure the HTTPS server without having to repeatedly type the **https server name** command:

```
ContentEngine(config-https)# cert testcert
ContentEngine(config-https)# key testkey
ContentEngine(config-https)# host www.mycompany-testserver.com
ContentEngine(config-https)# enable
```

The following example displays the output from the **show running-configuration** EXEC command entered on a Content Engine with the HTTPS server enabled:

```
ContentEngine# show running-config
!
!
wccp router-list 1 10.77.157.217
wccp https-cache router-list-num 1 password ****
wccp version 2
!
https server UNI certgroup chain uni-group
https server UNI certgroup serverauth uni-group
https server UNI key uni-key password ****
https server UNI host 10.77.157.170
https server UNI serverauth ignore domain-name
https server UNI enable
https server DSA210 certgroup chain dsa-group
https server DSA210 certgroup serverauth dsa-group
https server DSA210 key dsa-210-key password ****
https server DSA210 host 10.77.157.210
https server DSA210 serverauth ignore domain-name
https server DSA210 enable
https server test-w2k cert iis-crt
https server test-w2k key iis-key password ****
https server test-w2k host 10.77.140.131
no https server test-w2k serverauth enable
https server test-w2k session-cache size 20000
https server test-w2k session-cache timeout 86400
https server test-w2k enable
!
!
```

Related Commands

https (EXEC mode)
show bypass list
show https
show https server
show statistics https error
show statistics https requests
wccp https-cache
wccp router-list
wccp version

icap

To use the Internet Content Adaptation Protocol (ICAP) to help the Content Engine interact with third-party software applications and plug-ins, use the **icap** global configuration command. To disable individual options, use the **no** form of this command.



Note

See the “[icap service](#)” section for information about configuring specific ICAP services.

```
icap {append-x-headers {x-authenticated-groups | x-authenticated-user | x-client-ip |
x-server-ip} | apply {all | rules-template} | bypass streaming-media | connection-timeout
minutes | logging {enable | format standard} | rescan-cache IStag-change}
```

```
no icap {append-x-headers {x-authenticated-groups | x-authenticated-user | x-client-ip |
x-server-ip} | apply {all | rules-template} | bypass streaming-media | connection-timeout
minutes | logging {enable | format standard} | rescan-cache IStag-change}
```

Syntax Description

append-x-headers	Appends x-headers during the ICAP protocol handshake. Disabled by default. Can have multiple entries for various x-headers to be appended.
x-authenticated-groups	Appends x-authenticated-groups headers to the request that is sent to the ICAP server. Disabled by default.
x-authenticated-user	Appends x-authenticated-user headers to the request that is sent to the ICAP server. Disabled by default.
x-client-ip	Appends x-client-IP headers to the request that is sent to the ICAP server. Disabled by default.
x-server-ip	Appends x-server-IP headers to the request that is sent to the ICAP server. Disabled by default.
apply	Enables ICAP processing for HTTP requests.
all	Enables ICAP processing for all HTTP requests.
rules-template	Enables ICAP processing for HTTP requests that match the rule action use-icap-service global configuration command.
bypass	Enables bypassing of certain requested content.
streaming-media	Enables bypassing of streaming media.
connection-timeout	Sets the connection timeout for an ICAP transaction.
<i>minutes</i>	Number of minutes before the connection times out. The range is 1–480 minutes. The default is 20 minutes.
logging	Specifies logging-related options when using ICAP services.
enable	Enables logging when using ICAP services.
format	Specifies the format of the logging option chosen.
standard	Specifies the standard format as the logging option.
rescan-cache	Rescans cached objects.
IStag-change	Forces the strict rechecking of the cached content every time that the IStag changes.

Defaults

logging: ICAP transaction logging is disabled by default

bypass streaming-media: enabled

Command Modes

global configuration

Usage Guidelines

ICAP is an open standard for content adaptation, typically at the network edge. Content adaptation is the process of modifying the content to improve its usability. The content adaptation can include virus scanning, content translation, content filtering, content insertion, and other ways of improving the value of content for end users. ICAP specifies how a Content Engine, acting as an HTTP proxy server, can communicate with an external device acting as an ICAP server, which filters and adapts the requested content.

ICAP provides two content-processing modes for HTTP services. These modes define the transactions that can occur between a Content Engine acting as an ICAP client and an ICAP server. The two modes are as follows:

- Request modification (reqmod)—Allows modification of requests as they are sent from the Content Engine to the ICAP server on their way to the origin server. The ICAP server can modify these requests depending on the services requested.
- Response modification (respmod)—Allows modification of requests after they return from the origin server. The ICAP server only acts on requested objects, after they return from the origin server.

The following is a complete list of the ICAP vendors that have been certified to interoperate with the Content Engine:

- TrendMicro for reqmod and respmod
- Symantec for respmod

For ACNS 5.4.x software and later, the maximum file size that is supported in the ACNS software is 2 GB. Files that exceed this size limit are not supported for ICAP processing.

For releases prior to ACNS 5.4.x software, the maximum file size that is supported in the ACNS software in pass-through mode is 2 GB. Files that exceed this size limit are not supported for ICAP processing.

Use the **icap append-x-headers** global configuration command to specify the ICAP extension headers that are passed to the ICAP server during the session negotiation between the Content Engine and the ICAP server as follows:

- In the ACNS software, Release 5.1 and later releases, you can configure the Content Engine to append the client and server IP address headers to the request that is passed to the ICAP server. This capability allows you to use your ICAP server to perform URL filtering based on the client IP address and server IP address. To enable this capability, you must use the **icap append-x-headers x-client-ip** and **icap append-x-headers x-server-ip** command options.
- In the ACNS 5.2 software and later releases, you can configure the Content Engine to append the username and group name headers to the request that is passed to the ICAP server. This capability allows you to use your ICAP server to perform URL filtering based on username and group name.

To support this capability, two command options were added to the **icap append-x-headers** global configuration command:

- The **x-authenticated-user** option that allows the username information to be passed to the ICAP server for global services. This option is disabled by default. When this option is enabled (the **icap append-x-headers x-authenticated-user** option), the x-authenticated-user information is inserted into the ICAP request to the ICAP server.
- The **x-authenticated-groups** option that allows the group name information to be passed to the ICAP server for global services. This option is disabled by default. When this option is enabled (the **icap append-x-headers x-authenticated-groups** option), the x-authenticated-groups information is inserted into the ICAP request to the ICAP server.

The currently supported authentication schemes include LDAP, NTLM, RADIUS, and TACACS+.

To enable the ICAP transaction log, which is separate from the HTTP transaction log, use the **icap logging enable** command. You also have to enable transaction logging by entering the **transaction-log enable** command. The ICAP transaction log will be available in the /logs/icap directory. Only the standard logging format is supported in the ACNS software.

By default, streaming media requests are bypassed by ICAP processing. Use the **no icap bypass streaming-media** command to direct streaming media requests from Windows, RealMedia, and QuickTime media players through ICAP processing.

To configure a timeout value for ICAP connections, use the **icap connection-timeout minutes** global configuration command.

Also, you can choose to apply ICAP services on all HTTP requests processed by the Content Engine or apply ICAP processing only to requests that match the Rules Template. Use the **icap apply {all | rules-template}** global configuration command to specify which ICAP services should be performed on which requests that are received by the Content Engine. For example, use the **icap apply rules-template** global configuration command to instruct the Content Engine to run only the ICAP services that match the rules action **use-icap-service**. Alternatively, you could use the **icap apply all** global configuration command to instruct the Content Engine to run all of the ICAP services on all of the HTTP requests that it receives.

To exclude other traffic from ICAP processing, use the **rule action use-icap-service** command. The Rules Template can be used to turn off ICAP processing for various requests by applying the patterns available in the Rules Template to the incoming request. These patterns can include the following attributes of the incoming request:

- User-agent field
- Destination IP address
- Domain name

Use the **icap service** command to enter ICAP configuration mode and to configure a specific ICAP service.

Use the **icap rescan-cache IStag-change** command to force strict rechecking of the cached content every time that the IStag changes. The IStag is a field in the HTTP response header that allows ICAP servers to send a service-specific cookie to an ICAP client, representing the current state of the service. The IStag may change as a result of an update to the server version, virus pattern file, or policy. Changing or updating the IStag informs the ICAP client that previously cached responses on the client may no longer be fresh and should not be used.

After you enter this command, the Content Engine revalidates all cached content when there is a change in the IStag of the ICAP service. By default, cached content is *not* revalidated when the IStag changes.

Examples

The following example applies ICAP processing to all traffic:

```
ContentEngine(config)# icap apply all
```

The following examples exclude intranet traffic from ICAP processing:

```
ContentEngine(config)# rule pattern-list 1 domain !cisco.com
ContentEngine(config)# rule action use-icap-service trend-reqmod 1 protocol all
ContentEngine(config)# rule action use-icap-service trend-respmod 1 protocol all
ContentEngine(config)# rule enable
ContentEngine(config)# icap apply rules-template
```

The following example shows how to use the **icap service *service-id*** global configuration command to configure and enable various ICAP services on this Content Engine:

```
ContentEngine(config)# icap service trend-reqmod
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point reqmod-precache
ContentEngine(config-icap-service)# server icap//172.19.227.150/REQ-Service
ContentEngine# exit
```

```
ContentEngine(config)# icap service trend-respmod
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point respmod-precache
ContentEngine(config-icap-service)# server icap//172.19.227.150/interscan
ContentEngine# exit
```

Related Commands

```
icap service
rule use-icap-service
show icap
show icap service
show rule action use-icap-service
show statistics icap
```

icap service

To enter ICAP service configuration mode and configure ICAP services, enter the **icap service** global configuration command. To disable individual options, use the **no** form of this command.

icap service *service-id*

no icap service *service-id*

enable | **error-handling** { **bypass** | **return-error** } **exit** | **load-balancing** { **client-ip-hash** | **round-robin** | **server-ip-hash** | **weighted-load** } | **server** *url* [**keepalive-interval** *interval* [**max-connections** *number*] [**weight** *value*] | **max-connections** *number* [**keepalive-interval** *interval*] [**weight** *value*] | **weight** *value* [**keepalive-interval** *interval*] [**max-connections** *number*]]] | **vector-point** { **reqmod-postcache** | **reqmod-precache** | **respmod-precache** }

no enable | **error-handling** { **bypass** | **return-error** } | **load-balancing** { **client-ip-hash** | **round-robin** | **server-ip-hash** | **weighted-load** } | **server** *url* [**keepalive-interval** *interval* [**max-connections** *number*] [**weight** *value*] | **max-connections** *number* [**keepalive-interval** *interval*] [**weight** *value*] | **weight** *value* [**keepalive-interval** *interval*] [**max-connections** *number*]]] | **vector-point** { **reqmod-postcache** | **reqmod-precache** | **respmod-precache** }

Syntax Description

<i>service-id</i>	Name of your choice for the current ICAP service.
enable	Enables ICAP services.
error-handling	Specifies error-handling options when using an ICAP service.
bypass	Bypasses this service when an error occurs using this service.
return-error	Returns an error message to the client and ends the request.
	Note These errors are also entered in the transaction log to show the status of the action performed by the ICAP service.
exit	Exits from the ICAP service configuration mode.
load-balancing	Specifies a load-balancing option for this service.
client-ip-hash	Uses a hash-based algorithm based on the client IP address for load balancing the ICAP servers in the cluster.
round-robin	Uses the round-robin method in which ICAP servers take turns processing HTTP requests.
server-ip-hash	Uses a hash-based algorithm based on the server IP address for load balancing among the ICAP servers in the cluster.
weighted-load	Uses a farm of ICAP servers with different load capacities.
server	Enables the ICAP server to be used for ICAP services using a URL format.
<i>url</i>	URL based on the format <code>icap://ICAPserverIPaddress/service-name</code> .
keepalive-interval	(Optional) Sets the keepalive probe interval for this ICAP server.
<i>interval</i>	Keepalive probe interval in seconds (1–3600).
max-connections	(Optional) Specifies the maximum number of simultaneous connections to a particular ICAP server.
<i>number</i>	Maximum number of connections (1–5000).

weight	(Optional) Sets the weight for load balancing when you use the weighted-load option with the load balancing command.
value	Relative amount of traffic (0–100) assigned to the specified server.
vector-point	Configures when the specified ICAP service is applied to the client request or server response.
reqmod-postcache	Allows the modification of requests sent by the ICAP server and cached by the Content Engine.
reqmod-precache	Allows the modification of requests as they are sent from the Content Engine to the ICAP server on their way to the origin server.
respmod-precache	Allows the modification of responses before they are stored in the cache by the Content Engine.

Defaults

server url: port 1344 is assumed if no explicit port is included in the URL.

Command Modes

global configuration

Usage Guidelines

An ICAP service defines attributes that define the service and one or more servers that provide ICAP services. You can configure a maximum of ten ICAP services on a single Content Engine and a maximum of five ICAP servers for each ICAP service. To select the type of load balancing to use among a cluster of ICAP servers, use the **load-balancing** option.

In the syntax, replace *service-id* with a name of your choice for the current ICAP service. When you enter the **icap service** command and provide a name for the ICAP service, the system displays this ICAP service configuration prompt:

```
ContentEngine(config-icap-service)#
```

Within ICAP service configuration mode, all commands that you enter apply to the current ICAP service. To return to global configuration mode, enter the **exit** command. The following commands and options are available within ICAP service configuration mode:

The point at which ICAP services are applied to content is called the *vectoring point*, specified using the **vector-point** option. The following three vectoring points are supported:

- Client request vectoring point (**reqmod-precache**)—The ICAP server performs one of the following actions in response to the client request:
 - Terminates the connection
 - Sends a modified error response
 - Searches the cache using the URL in the request
 - Searches the cache using a modified URL
 - Modifies the request header or request body in the case of a cache miss
- Cache miss vectoring point (**reqmod-postcache**)—The ICAP server performs one of the following actions before forwarding the request to the origin server:
 - Terminates the connection
 - Sends a modified error response
 - Sends the request to the origin server using the original URL

- Sends the request to the origin server using an alternative URL
- Modifies the request header or request body
- Server response vectoring point (**respmod-precache**)—The ICAP server performs one of the following actions after receiving the response from the origin server:
 - Returns the response to the client
 - Modifies the request header or request body
 - Caches the response using the original URL
 - Caches the response using an alternative URL

With the **respmod** vectoring point, which is used by virus-scanning ICAP vendors, the performance of the Content Engine model CE-7305 will be 300 transactions per second.

With the **reqmod-precache** vectoring point, which is used by URL filtering ICAP vendors, the performance of the Content Engine model CE-7305 will drop 20 percent from the rated performance.

**Note**

The performance of the Content Engine will be limited by the performance of the ICAP server.

**Note**

Different ICAP services assigned to the same vectoring point can use different load-balancing options.

ICAP servers process HTTP requests from clients based on the ICAP services configured at various vectoring points. ICAP servers perform content adaptation such as a request or response modification and filtering of requests or responses at the configured vectoring points while processing HTTP requests.

You can configure the maximum number of connections and the weight that can be handled by an ICAP server in a cluster of servers. The weight parameter represents the load percentage that can be redirected to the ICAP server. An ICAP server with a weight of 40 denotes that this server handles 40 percent of the load. If the total weight of all ICAP servers in a load-balanced cluster exceeds 100, the load percentage for each ICAP server is recalculated as a percentage measure represented by the weight parameters.

**Note**

Always locate the ICAP server on a public LAN and configure its public IP address on the Content Engine. The ICAP server should not be located behind a NAT device.

ICAP servers configured at various vectoring points (especially the request modification precache vectoring points) may become overloaded with HTTP requests, because all requests pass through this point. Therefore, a cluster of ICAP servers (a load-balanced collection of ICAP servers) is made available for configuration. At a particular vectoring point, you can choose to load balance requests among the ICAP cluster of servers based on various parameters such as weighted load, client IP and server IP address-based hash, or round-robin format.

More than one ICAP service can be associated with a vectoring point. An ICAP service configured at a vectoring point can have only one load-balancing scheme, irrespective of the number of servers. However, multiple ICAP services configured at one or all of the vectoring points can have different load-balancing schemes.

To identify the specific ICAP server and service, use the **server url** command, where the URL is in the following format:

icap://ICAPserverIPaddress/service-name

The value used for the *service-name* must match the identifier used by the specific ICAP vendor. For example, one vendor uses the service name REQ-Service for **reqmod-precache** and interscan for **respmod-precache**, while another vendor supports only **respmod-precache** and uses the service name avscan.

When ICAP processing is enabled and an HTTP browser with a streaming Java applet is opened, several undesirable things occur as follows:

- The data for the Java applet is not updated in the browser. For example, when viewing a stock investment website, a user would not see any streaming stock updates.
- The ICAP daemon on the Content Engine continues to send updates (from the HTTP response) to the ICAP server, which overloads the ICAP server.

These conditions occur because the ICAP server is set up to inspect the entire data packet before delivering a response to the client. However, because there is a streaming request, the data continues flowing to the ICAP server indefinitely, deadlocking any response to the requesting client.

Two workarounds are available. You can configure the ICAP server to bypass the scanning process, or you can configure rules on the Content Engine to skip ICAP processing on websites that are known to contain streaming Java applets.

To configure the ICAP server to bypass scanning, use rules such as `client_skip_content` or `server_skip_content` as follows:

- The `client_skip_content` rule bypasses scanning on the basis of an HTTP request. The software looks for patterns in the HTTP header and bypasses all requests that exactly match the patterns specified in the `intscan.ini` file as follows:

```
client_skip_content=User Agent: Windows Media Player 9.0.1
```

- The `server_skip_content` rule bypasses scanning on the basis of an HTTP response. The software looks for patterns in the HTTP header and bypasses all responses that exactly match the patterns specified in the `intscan.ini` file as follows:

```
server_skip_content=Content-Type: X-Dave_Content
```

Alternatively, you can configure the Content Engine to bypass ICAP processing based on user agents or any of the patterns available in the Rules Template, by using the **rule** command.

Examples

The following examples show a typical configuration for a virus scanning service that requires processing on two vectoring points (**reqmod-precache** and **respmod-precache**):

```
ContentEngine(config)# icap apply all
ContentEngine(config)# icap service trend-reqmod
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point reqmod-precache
ContentEngine(config-icap-service)# server icap://172.19.227.150/REQ-Service
ContentEngine# exit
ContentEngine# icap service trend-respmod
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point respmod-precache
ContentEngine(config-icap-service)# server icap://172.19.227.150/interscan
ContentEngine# exit
```

The following example shows that if an ICAP vendor supports the same service name for more than one vectoring point, you can configure a single service and add the supported vectoring points:

```
ContentEngine(config)# icap service myicap-service
ContentEngine(config-icap-service)# enable
ContentEngine(config-icap-service)# vector-point reqmod-precache
```

```
ContentEngine(config-icap-service)# vector-point respmod-precache
ContentEngine(config-icap-service)# server icap://172.19.227.150/icap-service-name
ContentEngine(config-icap-service)# exit
ContentEngine(config)#
```

The following example shows that the Content Engine is configured to bypass ICAP processing on the intranet site cisco.com and on the trusted Internet site datek.com:

```
CE(config)# rule enable
CE(config)# rule action use-icap-service trend-reqmod pattern-list 1 protocol all
CE(config)# rule action use-icap-service trend-respmod pattern-list 1 protocol all
CE(config)# rule pattern-list 1 domain !(*cisco\.com|*datek\.com)!
CE(config)# icap apply rules-template
CE(config)# icap service trend-reqmod
CE(config-icap-service)# enable
CE(config-icap-service)# vector-point reqmod-precache
CE(config-icap-service)# server icap://172.19.227.150/REQ-Service
CE(config-icap-service)# exit
CE(config)# icap service trend-respmod
CE(config-icap-service)# enable
CE(config-icap-service)# vector-point respmod-precache
CE(config-icap-service)# server icap://172.19.227.150/interscan
CE(config-icap-service)# exit
```

Related Commands

```
icap
rule use-icap-service
show icap
show icap service
show rule action use-icap-service
show statistics icap
```


icp

To configure the Internet Cache Protocol (ICP) client and server, use the **icp** global configuration command. To disable the ICP client and server, use the **no** form of this command.

```

icp client add-remote-server {hostname | ip-address} {parent | sibling} icp-port icpport
http-port httpport [restrict domainnames]

icp client enable

icp client exclude domainnames

icp client max-fail retries

icp client max-wait timeout

icp client modify-remote-server {hostname | ip-address} {http-port port | icp-port port | parent |
restrict domainnames | sibling}

icp server enable

icp server http-port port

icp server port icpport

icp server remote-client {hostname | ip-address} {fetch | no-fetch}

no icp {client {{add-remote-server {hostname | ip-address} | enable | exclude domainnames |
max-fail | max-wait | modify-remote-server {hostname | ip-address} {http-port port |
icp-port port | parent | restrict domainnames | sibling}} | server {enable | http-port | port |
remote-client {hostname | ip-address}}}
```

Syntax Description

client	Sets the ICP client functionality.
add-remote-server	Adds an ICP client remote server.
<i>hostname</i>	Hostname of remote server.
<i>ip-address</i>	IP address of remote server.
parent	Specifies the ICP server that acts like a parent.
sibling	Specifies the ICP server that acts like a sibling.
icp-port	Sets the ICP port to receive remote requests.
<i>icpport</i>	ICP port number (0–65535).
http-port	Sets the HTTP port to receive HTTP requests.
<i>httpport</i>	HTTP request port number (0–65535).
restrict	(Optional) Sets a restricted list of domains.
<i>domainnames</i>	Space-delimited restricted domain list.
enable	Enables the ICP client.
exclude	Excludes ICP client local domains.
<i>domainnames</i>	Space-delimited local domain list.
max-fail	Sets the maximum number of retries allowed.

<i>retries</i>	Number of retries (0–100).
max-wait	Sets the maximum wait for ICP responses before the timeout occurs.
<i>timeout</i>	Timeout period for ICP responses in seconds (0–30).
modify-remote-server	Modifies the ICP client remote server parameters.
<i>hostname</i>	Hostname of remote server.
<i>ip-address</i>	IP address of remote server.
http-port	Sets the HTTP port.
<i>port</i>	HTTP request port number (0–65535).
icp-port	Sets the ICP port.
<i>port</i>	ICP request port number (0–65535).
parent	Sets the ICP remote server that acts like a parent.
restrict	Sets the restricted list of domains.
<i>domainnames</i>	Space-delimited local domain list.
sibling	Sets the ICP remote server that acts like a sibling.
server	Sets the ICP server functionality.
enable	Enables the ICP server.
http-port	Sets the HTTP proxy port to listen for ICP-generated requests.
<i>port</i>	HTTP server port number (0–65535) for ICP.
port	Sets the ICP server listener port that listens for ICP requests.
<i>icpport</i>	ICP request port number (0–65535).
remote-client	Sets the ICP server remote client.
<i>hostname</i>	Hostname of remote client.
<i>ip-address</i>	IP address of remote client.
fetch	Sets the ICP remote client to fetch a cache miss.
no-fetch	Sets the ICP remote client to not fetch a cache miss.

Defaults**max-wait:** 2 seconds**max-fail:** 20 failures**icp-port:** 3130**http-port:** 3128**Command Modes**

global configuration

Usage Guidelines

ICP is a message format used for communicating among Content Engines and for supporting interoperability with older proxy protocols. ICP is used to exchange hints about the existence of URLs in neighboring Content Engines in a Content Engine cluster (farm). Content Engines exchange ICP queries and replies to gather information for use in selecting the most appropriate location from which to retrieve an object.

Although ICP has traditionally been a way to scale the overall size of a cluster of Content Engines beyond a single unit, using ICP is a poor choice for scaling a Content Engine clustering solution. Because of the way that traffic is currently directed toward a transparent network Content Engine cluster, the requirement for ICP is ineffective for the majority of Content Engine deployments.

The ICPv2 protocol is documented in two standards documents:

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

You can configure your Content Engine cluster to generate ICP queries before retrieving requested objects from the Internet using ICP client functionality. With ICP, you can configure parent and sibling Content Engines in a caching hierarchy. ICP parents are essentially one step higher than ICP siblings in a hierarchy of Content Engines.

You can configure a Content Engine to be either a parent or a sibling. Parent Content Engines are able to retrieve data during a cache miss. Sibling Content Engines cannot retrieve data and instead forward the request to the parent Content Engines.

You can also configure a Content Engine to act as an ICP server. This configuration allows the Content Engine to probe the hierarchy of Content Engines by multicasting an ICP message to ICP parent and sibling clients in the hierarchy.

Use the **icp** global configuration commands to establish and configure the ICP server and client functionality of the Content Engine. Configurations made without enabling ICP functionality are stored within the configuration until removed.

To enable ICP server or client functionality, use the **icp {server | client} enable** command. Be sure to enable ICP on any other Content Engines or ICP servers or clients within the ICP environment. You can monitor the statistical data of the ICP service using the **show statistics icp EXEC** command.

Examples

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

```
ContentEngine(config)# icp client add-remote-server 172.16.0.0 parent icp-port 3130  
http-port 3128 domain_x.com domain_y.com domain_z.com
```

```
ContentEngine(config)# icp client add-remote-server 172.16.0.0 sibling icp-port 3130  
http-port 3128 domain_a.com domain_b.com domain_c.com
```

The following example enables an ICP client on the Content Engine:

```
ContentEngine(config)# icp client enable  
Icp Client started
```

Related Commands

```
show icp client  
show icp server  
show statistics icp
```

inetd

To enable FTP, RCP, and TFTP services, use the **inetd** global configuration command. To disable these same services, use the **no** form of this command.

inetd enable {ftp | rcp | tftp}

no inetd enable {ftp | rcp | tftp}

Syntax Description

enable	Enables services.
ftp	Enables FTP services.
rcp	Enables RCP services.
tftp	Enables TFTP services.

Defaults

FTP is enabled by default; RCP and TFTP are disabled.

Command Modes

global configuration

Usage Guidelines

Inetd is a program that listens for connection requests or messages for certain ports and starts server programs to perform the services associated with those ports. Use the **inetd enable** command with the **ftp**, **rcp**, and **tftp** keywords to enable and disable services on the Content Engine. To disable the service, enter the **no** form of the **inetd enable** command. Use the **show inetd** command to see whether current **inetd** sessions are enabled or disabled.

Remote Copy Program (RCP) provides a method of downloading, uploading, and copying configuration files between remote hosts and the switch. Unlike TFTP, which uses User Datagram Protocol (UDP), a connectionless protocol, RCP uses Transmission Control Protocol (TCP), which is connection oriented.

RCP, which is a subset of the UNIX rshell service, allows UNIX users to execute shell commands on remote UNIX systems. It is a UNIX built-in service. This service uses Transmission Control Protocol (TCP) as the transport protocol and listens for requests at TCP port 514. RCP service can be enabled on Content Engines running the ACNS software.

In the ACNS software 5.3 and later releases, enabling the SSH daemon also enables the Secure File Transfer Protocol (SFTP) server.

Examples

The following example enables an FTP service session:

```
ContentEngine(config)# inetd enable ftp
```

The following example disables FTP services:

```
ContentEngine(config)# no inetd enable ftp
```


Related Commands

show inetd
sshd

install

To install the ACNS software image, use the **install** EXEC command.

install *imagefilename*

Syntax Description	<i>imagefilename</i> Name of the .bin file that you want to install.
Defaults	No default behavior or values
Command Modes	EXEC
Usage Guidelines	<p>The install command loads the system image into flash memory and the disk.</p> <p>To install a system image, copy the image file to the sysfs directory local1 or local2. Before entering the install command, change the present working directory to the directory where the system image resides. When the install command is executed, the image file is expanded. The expanded files overwrite the existing files in the Content Engine. The newly installed version takes effect after the system image is reloaded.</p>
 Note	The install command does not accept .pax files. Files should be of the type .bin (for example, ACNS-5.2.1.7-K9.bin). Also, if the release being installed does not require a new system image, then it may not be necessary to write to flash memory. If the newer version has changes that require a new system image to be installed, then the install command may result in a write to flash memory.
Examples	<p>The following example shows how to install a .bin file on the Content Engine:</p> <pre>ContentEngine# install ACNS-5.2.1.7-K9.bin</pre>
Related Commands	<p>copy ftp install copy http install reload</p>

interface

To configure a Fast Ethernet, Fibre Channel, Gigabit Ethernet, or port-channel interface, use the **interface** global configuration command. To disable selected options, restore default values, or enable a shut down interface, use the **no** form of this command.

```
interface FastEthernet slot/port [autosense | bandwidth {10 | 100 | 1000} | cdp enable |
channel-group {1 | 2} | description text | full duplex | half duplex | ip {access-group
{acl-num | acl_name} {in | out} | address {ip_address netmask [secondary] | dhcp [client-id
id hostname name | hostname name client-id id}} | mtu mtusize | shutdown | standby
grpnumber [priority priority]]
```

```
interface FibreChannel slot/port [description text | mode {autosense | direct-attached |
switched} | speed {1 | 2 | autosense}]
```

```
interface GigabitEthernet slot/port [autosense | bandwidth {10 | 100 | 1000} | cdp enable |
channel-group {1 | 2} | description text | full duplex | half duplex | ip {access-group
{acl-num | acl_name} {in | out} | address {ip_address netmask [secondary] | dhcp [client-id
id hostname name | hostname name client-id id}} | mtu mtusize | shutdown | standby
grpnumber [priority priority]]
```

```
interface PortChannel {1 | 2} [description text | ip {access-group {acl-num | acl_name} {in | out}
| address ip-address netmask} | shutdown]
```

```
no interface {FastEthernet slot/port [autosense | bandwidth {10 | 100 | 1000} | cdp enable |
channel-group {1 | 2} | description | full duplex | half duplex | ip {access-group {acl-num |
acl_name} {in | out} | address {ip_address netmask [secondary] | dhcp [client-id id hostname
name | hostname name client-id id}} | mtu mtusize | shutdown | standby grpnumber
[priority priority]] | FibreChannel slot/port [description | mode {autosense | direct-attached
| switched} | speed {1 | 2 | autosense}}] | GigabitEthernet slot/port [autosense | bandwidth
{10 | 100 | 1000} | cdp enable | channel-group {1 | 2} | description | full duplex | half duplex
| ip {access-group {acl-num | acl_name} {in | out} | address {ip_address netmask [secondary]
| dhcp [client-id id hostname name | hostname name client-id id}} | mtu mtusize | shutdown
| standby grpnumber [priority priority]] | PortChannel {1 | 2} [description | ip
{access-group {acl-num | acl_name} {in | out} | address ip-address netmask} | shutdown}]
```

Syntax Description

FastEthernet	Selects a Fast Ethernet interface to configure.
<i>slot/port</i>	Slot and port number for the selected interface. The slot range is 0–3; the port range is 0–3. The slot number and port number are separated with a forward slash character (/).
autosense	(Optional) Sets the interface to autosense.
bandwidth	(Optional) Sets the bandwidth of the specified interface.
10	Sets the bandwidth of the interface to 10 megabits per second (Mbps).
100	Sets the bandwidth of the interface to 100 Mbps.
1000	Sets the bandwidth of the interface to 1000 Mbps. This option is not available on all ports and is the same as autosense.
cdp	(Optional) Configures Cisco Discovery Protocol on the specified interface.
enable	Enables Cisco Discovery Protocol on the specified interface.

channel-group	(Optional) Configures the EtherChannel group.
1	Assigns the interface's EtherChannel to group 1.
2	Assigns the interface's EtherChannel to group 2.
description	(Optional) Sets the description for the specified interface.
<i>text</i>	Description for the specified interface. The maximum length of the description text is 240 characters.
full duplex	(Optional) Sets the interface to full-duplex operation.
half duplex	(Optional) Sets the interface to half-duplex operation.
ip	(Optional) Enables IP configuration commands for the interface.
access-group	Configures access control for IP packets on this interface using an access control list (ACL).
<i>acl-num</i>	Numeric identifier that identifies the ACL to apply to the current interface. For standard ACLs, the valid range is 1–99; for extended ACLs, the valid range is 100–199.
<i>acl_name</i>	Alphanumeric identifier of up to 30 characters, beginning with a letter that identifies the ACL to apply to the current interface.
in	Applies the specified ACL to inbound packets on the current interface.
out	Applies the specified ACL to outbound packets on the current interface.
address	(Optional) Sets the interface IP address and subnet mask.
<i>ip-address</i>	IP address of the interface.
<i>netmask</i>	Netmask of the interface.
secondary	(Optional) Makes this IP address a secondary address.
dhcp	(Optional) Sets the IP address to that negotiated over Dynamic Host Configuration Protocol (DHCP).
client-id	(Optional) Specifies the client identifier.
<i>id</i>	Client identifier.
hostname	(Optional) Specifies the hostname.
<i>name</i>	Hostname.
mtu	(Optional) Sets the interface Maximum Transmission Unit (MTU) size.
<i>mtusize</i>	MTU size in bytes (68–1500).
shutdown	(Optional) Shuts down this interface.
standby	(Optional) Sets standby interface configuration commands.
<i>grpnumber</i>	Standby group number (1–4).
priority	(Optional) Sets the priority of an interface for the standby group.
<i>priority</i>	Interface priority for the standby group (0–4294967295).
FibreChannel	Selects the Fibre Channel interface to configure.
<i>slot/port</i>	Slot and port number for the selected interface. The slot range is 0–0; the port range is 0–3. The slot number and port number are separated with a forward slash character (/).
mode	(Optional) Sets the Fibre Channel interface operation mode. For more information, see the “mode” section on page 2-312 .
autosense	Sets the operation mode of the Content Engine to autosense.
direct-attached	Sets the operation mode when the Content Engine is directly connected to a storage array.

switched	Sets the operation mode when the Content Engine is connected to a switch.
speed	(Optional) Sets the Fibre Channel interface speed.
1	Sets the Fibre Channel interface speed to 1 gigabit per second (Gbps).
2	Sets the Fibre Channel interface speed to 2 Gbps.
autosense	Sets the Fibre Channel to automatically sense the interface speed.
GigabitEthernet	Selects a Gigabit Ethernet interface to configure.
<i>slot/port</i>	Slot and port number for the selected interface. The slot range is 0–2; the port range is 0–3. The slot number and port number are separated with a forward slash character (/).
PortChannel	Selects the EtherChannel of interfaces to configure.
1	Sets the port-channel interface number to 1.
2	Sets the port-channel interface number to 2.
ip	(Optional) Enables IP configuration commands for the interface.
access-group	Configures access control for IP packets on this interface using access control list (ACL).
<i>acl_num</i>	Numeric identifier that identifies the ACL to apply to the current interface. For standard ACLs, the valid range is 1–99; for extended ACLs, the valid range is 100–199.
<i>acl_name</i>	Alphanumeric identifier of up to 30 characters, beginning with a letter that identifies the ACL to apply to the current interface.
in	Applies the specified ACL to inbound packets on the current interface.
out	Applies the specified ACL to outbound packets on the current interface.
address	Sets the interface IP address.
<i>ip-address</i>	IP address of this interface.
<i>netmask</i>	Netmask of this interface.
shutdown	(Optional) Shuts down this interface.

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

To configure an interface bandwidth on a Content Engine, use the **bandwidth** interface configuration command. The bandwidth is specified in megabits per second (Mbps). The **1000** Mbps option is not available on all ports. Using this option automatically enables autosense on the interface. You cannot change the interface speed on a Content Engine CE-7320 model that has an optical Gigabit Ethernet interface. Therefore, Gigabit Ethernet interfaces only run at 1000 Mbps for a CE-7320. For newer models of the Content Engine (for example, the CE-510, CE-565, CE-7305, CE-7325, and CE-7326) that have a Gigabit Ethernet interface over copper, this restriction does not apply; you can configure these Gigabit Ethernet interfaces to run at 10, 100, or 1000 Mbps. On these newer Content Engine models, the 1000-Mbps setting implies autosense (for example, you cannot configure the Gigabit Ethernet interface to run at 1000 Mbps and half duplex).

Using the **cdp enable** command in global configuration mode enables CDP globally on all the interfaces. If you want to control CDP behavior per interface, then use the **cdp enable** command in interface configuration mode. The interface level control overrides the global control.

To display the interface identifiers (for example, interface FastEthernet 0/1), use the **show running-config** or **show startup-config** commands. The **autosense**, **bandwidth**, **full duplex**, **half duplex**, **ip**, and **shutdown** commands are listed separately in this command reference.

Configuring Multiple Secondary IP Addresses on a Single Physical Interface

Use the **interface secondary** global configuration command to configure more than one IP address on the same interface. By configuring multiple IP addresses on a single interface, the Content Engine can be present in more than one subnet. This configuration allows you to optimize the response time because the content goes directly from the Content Engine to the requesting client without being redirected through a router. The Content Engine becomes visible to the client because both are configured on the same subnet.

Up to four secondary addresses can be assigned to an interface on a Content Engine. These addresses become active only after the primary address is configured. No two interfaces can have the same IP address in the same subnetwork. To set these secondary IP addresses, use the **ip address** command.

If a Content Engine has one physical interface that has multiple secondary IP addresses assigned to it, the egress traffic uses the source IP address that is chosen by IP routing. If the secondary IP addresses are in the same subnet as the primary IP address, then the egress traffic only uses the primary IP address. In contrast, if the secondary IP addresses are in a different subnet than the primary IP address, then the destination IP address determines which IP address on the Content Engine is used for the egress traffic.

Configuring Interfaces for DHCP

During the initial configuration of a Content Engine, you have the option of configuring a static IP address for the Content Engine or using interface-level DHCP to dynamically assign IP addresses to the interfaces on the Content Engine.

If you do not enable interface-level DHCP on the Content Engine, you must manually specify a static IP address and network mask for the Content Engine. If the Content Engine moves to another location in another part of the network, you must manually enter a new static IP address and network mask for this Content Engine.

An interface can be enabled for DHCP by using the **ip address dhcp [client_id | hostname]** interface configuration command. The client identifier is an ASCII value. The Content Engine sends its configured client identifier and hostname to the DHCP server when requesting network information. DHCP servers can be configured to identify the client identifier information and the hostname information that the Content Engine is sending and then send back the specific network settings that are assigned to the Content Engine.



Note

You must disable autoregistration before you can manually configure an interface for DHCP. Autoregistration is enabled by default on the first interface of the device.

Defining Interface Descriptions

In the ACNS 5.3 software and later releases, you can specify a one-line description for a specific interface on a Content Engine. You use the **description text** interface configuration command to enter the description for the specific interface. The maximum length of the description text is 240 characters. This feature is supported for the Fast Ethernet, Gigabit Ethernet, Fibre Channel, port-channel, and standby interfaces.

**Note**

This feature is not currently supported for the SCSI or IDE interfaces.

After you define the description for an interface, use the **show EXEC** commands to display the defined interface descriptions. Enter the **show interface interface type slot/port EXEC** command to display the defined description for a specific interface on the Content Engine.

Fibre Channel Interface

The ACNS 5.x software supports Fibre Channel interfaces. Fibre Channel is the chosen technology for interconnecting storage devices and servers in a storage area network (SAN). In a SAN, the storage does not need to be directly attached to the server, and data transfer occurs over a high-throughput, high-availability network. Fibre Channel can operate at speeds of 1 gigabit per second (Gbps) and 2 Gbps.

To detect the presence of Fibre Channel storage, the storage array must be configured to assign storage space for the Content Engine, and the Content Engine must be reloaded before it can detect the storage assignment. To confirm whether the Content Engine has detected the storage assignment, use the **show disks** and the **show disks details EXEC** commands.

To configure the Fibre Channel interface on the Content Engine, use the **interface FibreChannel slot/port** command in interface configuration mode.

Port-Channel (EtherChannel) Interface

EtherChannel for the ACNS 5.x software supports the grouping of up to four same-speed network interfaces into one virtual interface. This grouping allows the setting or removing of a virtual interface that consists of two, three, or four Fast Ethernet interfaces or two Gigabit Ethernet interfaces. EtherChannel also provides interoperability with Cisco routers, switches, and other networking devices or hosts supporting EtherChannel, load balancing, and automatic failure detection and recovery based on each interface's current link status.

You can use either the Fast Ethernet or the Gigabit Ethernet ports to form an EtherChannel; however, an EtherChannel cannot contain both Fast Ethernet and Gigabit Ethernet interfaces. A physical interface can be added to an EtherChannel subject to the device configuration.

Examples

The following example configures an attribute of an interface with a single CLI command:

```
ContentEngine(config)# interface FastEthernet 0/1 half-duplex
```

The following example shows that an interface can be configured in a sequence of CLI commands:

```
ContentEngine(config)# interface FastEthernet 0/1
ContentEngine(config-if)# half-duplex
ContentEngine(config-if)# exit
ContentEngine(config)#
```

The following example shows how to shut down an interface:

```
ContentEngine(config)# no interface FastEthernet 0/1 shutdown
```

The following example creates an EtherChannel. The port channel is port channel 2 and is assigned an IP address of 10.10.10.10 and a netmask of 255.0.0.0:

```
ContentEngine# configure
ContentEngine(config)# interface PortChannel 2
ContentEngine(config-if)# ip address 10.10.10.10 255.0.0.0
ContentEngine(config-if)# exit
```

The following example removes an EtherChannel:

```
ContentEngine(config)# interface PortChannel 2
ContentEngine(config-if)# no ip address 10.10.10.10 255.0.0.0
ContentEngine(config-if)# exit
ContentEngine(config)# no interface PortChannel 2
```

The following example adds an interface to a channel group:

```
ContentEngine# configure
ContentEngine(config)# interface FastEthernet 1/1
ContentEngine(config-if)# channel-group 2
ContentEngine(config-if)# exit
```

The following example removes an interface from a channel group:

```
ContentEngine(config)# interface FastEthernet 1/1
ContentEngine(config-if)# no channel-group 2
ContentEngine(config-if)# exit
```

The following example assigns a secondary IP address on a Fast Ethernet interface on a Content Engine using the **ip address** configuration interface command:

```
ContentEngine# configure
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# ip address 10.10.10.10 255.0.0.0 secondary
```

The following example configures a description for a Fast Ethernet interface:

```
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# description This is a FastEthernet interface.
```

The following example shows a sample output of the **show running-config EXEC** command:

```
ContentEngine# show running-config
.
.
.
interface FastEthernet 0/0
  description This is an interface to the WAN
  ip address dhcp
  ip address 192.168.1.200 255.255.255.0
  no autosense
  bandwidth 100
  full-duplex
  exit
.
.
.
```

The following example shows the sample output of the **show interface** command:

```
ContentEngine# show interface GigabitEthernet 1/0
Description: This is the interface to the lab
type: Ethernet
.
.
.
```

Related Commands

show interface
show running-config
show startup-config

ip

To change initial network device configuration settings, use the **ip** global configuration command. To delete or disable these settings, use the **no** form of this command. The **dscp** option allows you to set the global Type of Service (ToS) or differentiated services code point (DSCP) values in IP packets.

ip default-gateway *ip-address*

ip domain-name *name1 name2 name3*

ip dscp { **client** { **cache-hit** { **match-server** | **set-dscp** *dscp-packets* | **set-tos** *tos-packets* } | **cache-miss** { **match-server** | **set-dscp** *dscp-packets* | **set-tos** *tos-packets* } } | **server** { **match-client** | **set-dscp** *dscp-packets* | **set-tos** *tos-packets* } }

ip name-server *ip-addresses*

ip path-mtu-discovery **enable**

ip route *dest_addrs net_addrs gateway_addrs*

no ip { **default-gateway** | **domain-name** | **dscp** { **client** { **cache-hit** | **cache-miss** } | **server** } | **name-server** *ip-addresses* | **path-mtu-discovery** **enable** | **route** *dest_addrs net_addrs gateway_addrs* }

Syntax Description

default-gateway	Specifies the default gateway (if not routing IP).
<i>ip-address</i>	IP address of the default gateway.
domain-name	Specifies domain names.
<i>name1</i> through <i>name3</i>	Domain name (up to three can be specified).
dscp	Configures IP differentiated services code point (DSCP) and Type of Service (ToS) fields.
client	Configures DSCP for responses to the client.
cache-hit	Configures the cache hit responses to the client.
cache-miss	Configures the cache miss responses to the client.
match-server	Uses the original ToS/DSCP value of the server.
set-dscp	Configures differentiated services code point (DSCP) values.
<i>dscp-packets</i>	DSCP values; see Table 2-8 on page 2-270 for valid values.
set-tos	Configures Type of Service (ToS).
<i>tos-packets</i>	ToS value; see Table 2-10 on page 2-274 for valid values.
server	Configures DSCP for outgoing requests.
match-client	Uses the original ToS/DSP value of the client.
name-server	Specifies the address of the name server.
<i>ip-addresses</i>	IP addresses of the name servers (up to a maximum of eight).
path-mtu-discovery	Configures RFC 1191 Path Maximum Transmission Unit (MTU) discovery.
enable	Enables Path MTU discovery.
route	Specifies the net route.

<i>dest_addr</i>	Destination route address.
<i>net_addr</i>	Netmask address.
<i>gateway_addr</i>	Gateway address.

Defaults

No default behavior or values

Command Modes

global configuration

Usage Guidelines

To define a default gateway, use the **ip default-gateway** command. Only one default gateway can be configured. To remove the IP default gateway, use the **no** form of this command. The Content Engine uses the default gateway to route IP packets when there is no specific route found to the destination.

To define a default domain name, use the **ip domain-name** command. To remove the IP default domain name, use the **no** form of this command. Up to three domain names can be entered. If a request arrives without a domain name appended in its hostname, the proxy tries to resolve the hostname by appending *name1*, *name2*, and *name3* in that order until one of these names succeeds.

The Content Engine appends the configured domain name to any IP hostname that does not contain a domain name. The appended name is resolved by the DNS server and then added to the host table. The Content Engine must have at least one domain name server specified for hostname resolution to work correctly.

To specify the address of one or more name servers to use for name and address resolution, use the **ip name-server ip-addresses** command. To disable IP name servers, use the **no** form of this command. For proper resolution of the hostname to the IP address or the IP address to the hostname, the Content Engine uses DNS servers. Use the **ip name-server** command to point the Content Engine to a specific DNS server. You can configure up to eight servers.

Path MTU autodiscovery discovers the MTU and automatically sets the correct value. Use the **ip path-mtu-discovery enable** command to start this autodiscovery utility. By default, this feature is enabled. When this feature is disabled, the sending device uses a packet size that is smaller than 576 bytes and the next hop MTU. Existing connections are not affected when this feature is turned on or off.

The Cisco ACNS software supports IP Path MTU Discovery, as defined in RFC 1191. When enabled, Path MTU Discovery discovers the largest IP packet size allowable between the various links along the forwarding path and automatically sets the correct value for the packet size. By using the largest MTU that the links will bear, the sending device can minimize the number of packets that it must send.

**Note**

IP Path MTU Discovery is useful when a link in a network goes down, forcing the use of another, different MTU-sized link. IP Path MTU Discovery is also useful when a connection is first being established and the sender has no information at all about the intervening links.

IP Path MTU Discovery is initiated by the sending device. If a server does not support IP Path MTU Discovery, the receiving device will have no mechanism available to avoid fragmenting datagrams generated by the server.

Use the **ip route** command to add a specific static route for a network or host. Any IP packet designated for the specified destination uses the configured route.

To configure static IP routing, use the **ip route** command. To remove the route, use the **no** form of this command. Do not use the **ip route 0.0.0.0 0.0.0.0** command to configure the default gateway; use the **ip default-gateway** command instead.

In the ACNS network, you can configure Content Engines, Content Routers, and Content Distribution Managers for the Type of Service (ToS) or differentiated services code point (DSCP) using the **ip dscp** command.

Differentiated Services

The differentiated services (DiffServ) architecture is based on a simple model where traffic entering a network is classified and possibly conditioned at the boundaries of the network. The class of traffic is then identified with a differentiated services (DS) code point or bit marking in the IP header. Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DS code point.

To set the global ToS or DSCP values for the IP header from the CLI, use the **ip dscp** command.

DiffServ describes a set of end-to-end QoS (Quality of Service) capabilities. End-to-end QoS is the ability of the network to deliver service required by specific network traffic from one end of the network to another. QoS in the ACNS software supports differentiated services.

With differentiated services, the network tries to deliver a particular kind of service based on the QoS specified by each packet. This specification can occur in different ways, for example, using the 6-bit DSCP setting in IP packets or source and destination addresses. The network uses the QoS specification to classify, mark, shape, and police traffic, and to perform intelligent queueing.

Differentiated services is used for several mission-critical applications and for providing end-to-end QoS. Typically, differentiated services is appropriate for aggregate flows because it performs a relatively coarse level of traffic classification.

Use the **ip dscp {client | server} {cache-hit | cache-miss} set-dscp dscp-packets** command to set the DSCP values for the IP header. Valid values for *dscp-packets* are listed in [Table 2-8](#).

Table 2-8 *dscp-packets Values*

Value or Keyword	Description ¹
0–63	Sets DSCP values.
af11	Sets packets with AF11 DSCP (001010).
af12	Sets packets with AF12 DSCP (001100).
af13	Sets packets with AF13 DSCP (001110).
af21	Sets packets with AF21 DSCP (010010).
af22	Sets packets with AF22 DSCP (010100).
af23	Sets packets with AF23 DSCP (010110).
af31	Sets packets with AF31 DSCP (011010).
af32	Sets packets with AF32 DSCP (011100).
af33	Sets packets with AF33 DSCP (011110).
af41	Sets packets with AF41 DSCP (100010).
af42	Sets packets with AF42 DSCP (100100).
af43	Sets packets with AF43 DSCP (100110).
cs1	Sets packets with CS1 (precedence 1) DSCP (001000).

Table 2-8 *dscp-packets Values (continued)*

Value or Keyword	Description ¹
cs2	Sets packets with CS2 (precedence 2) DSCP (010000).
cs3	Sets packets with CS3 (precedence 3) DSCP (011000).
cs4	Sets packets with CS4 (precedence 4) DSCP (100000).
cs5	Sets packets with CS5 (precedence 5) DSCP (101000).
cs6	Sets packets with CS6 (precedence 6) DSCP (110000).
cs7	Sets packets with CS7 (precedence 7) DSCP (111000).
default	Sets packets with the default DSCP (000000).
ef	Sets packets with EF DSCP (101110).

1. The number in parentheses denotes the DSCP value for each per-hop behavior keyword.

DS Field Definition

A replacement header field, called the DS field, is defined by differentiated services. The DS field supersedes the existing definitions of the IPv4 ToS octet (RFC 791) and the IPv6 traffic class octet. Six bits of the DS field are used as the DSCP to select the Per Hop Behavior (PHB) at each interface. A currently unused (CU) 2-bit field is reserved for explicit congestion notification (ECN). The value of the CU bits is ignored by DS-compliant interfaces when determining the PHB to apply to a received packet.

Per-Hop Behaviors

RFC 2475 defines PHB as the externally observable forwarding behavior applied at a DiffServ-compliant node to a DiffServ Behavior Aggregate (BA).

With the ability of the system to mark packets according to the DSCP setting, collections of packets that have the same DSCP setting and that are sent in a particular direction can be grouped into a BA. Packets from multiple sources or applications can belong to the same BA.

A PHB refers to the packet scheduling, queueing, policing, or shaping behavior of a node on any given packet belonging to a BA, as configured by a service level agreement (SLA) or a policy map.

There are four available standard PHBs as follows:

- Default PHB (as defined in RFC 2474)
- Class-Selector PHB (as defined in RFC 2474)
- Assured Forwarding (AFny) PHB (as defined in RFC 2597)
- Expedited Forwarding (EF) PHB (as defined in RFC 2598)

The following sections describe the PHBs.

Default PHB

The default PHB specifies that a packet marked with a DSCP value of 000000 (recommended) receives the traditional best-effort service from a DS-compliant node (a network node that complies with all of the core DiffServ requirements). Also, if a packet arrives at a DS-compliant node, and the DSCP value is not mapped to any other PHB, the packet gets mapped to the default PHB.

Class-Selector PHB

To preserve backward compatibility with any IP precedence scheme currently in use on the network, DiffServ has defined a DSCP value in the form xxx000, where x is either 0 or 1. These DSCP values are called Class-Selector Code Points. (The DSCP value for a packet with default PHB 000000 is also called the Class-Selector Code Point.)

The PHB associated with a Class-Selector Code Point is a Class-Selector PHB. These Class-Selector PHBs retain most of the forwarding behavior as nodes that implement IP precedence-based classification and forwarding.

For example, packets with a DSCP value of 110000 (the equivalent of the IP precedence-based value of 110) have preferential forwarding treatment (for scheduling, queueing, and so on), as compared to packets with a DSCP value of 100000 (the equivalent of the IP precedence-based value of 100). These Class-Selector PHBs ensure that DS-compliant nodes can coexist with IP precedence-based nodes.

Assured Forwarding PHB

Assured Forwarding PHB is nearly equivalent to Controlled Load Service, which is available in the integrated services model. AFny PHB defines a method by which BAs can be given different forwarding assurances.

For example, network traffic can be divided into the following classes:

- Gold—Traffic in this category is allocated 50 percent of the available bandwidth.
- Silver—Traffic in this category is allocated 30 percent of the available bandwidth.
- Bronze—Traffic in this category is allocated 20 percent of the available bandwidth.

The AFny PHB defines four AF classes: AF1, AF2, AF3, and AF4. Each class is assigned a specific amount of buffer space and interface bandwidth according to the SLA with the service provider or policy map.

Within each AF class, you can specify three drop precedence (dP) values: 1, 2, and 3. Assured Forwarding PHB can be expressed as shown in the following example: AFny. In this example, n represents the AF class number (1, 2, or 3) and y represents the dP value (1, 2, or 3) within the AFn class.

In instances of network traffic congestion, if packets in a particular AF class (for example, AF1) need to be dropped, packets in the AF1 class will be dropped according to the following guideline:

$$dP(AFny) \geq dP(AFnz) \geq dP(AFnx)$$

where dP (AFny) is the probability that packets of the AFny class will be dropped and y denotes the dP within an AFn class.

In the following example, packets in the AF13 class will be dropped before packets in the AF12 class, which in turn will be dropped before packets in the AF11 class:

$$dP(AF13) \geq dP(AF12) \geq dP(AF11)$$

The dP method penalizes traffic flows within a particular BA that exceed the assigned bandwidth. Packets on these offending flows could be re-marked by a policer to a higher drop precedence.

An AFx class can be denoted by the DSCP value, xyzab0, where xyz can be 001, 010, 011, or 100, and ab represents the dP value.

Table 2-9 lists the DSCP value and corresponding dP value for each AF PHB class.

Table 2-9 DSCP Values and Corresponding Drop Precedence Values for Each AF PHB Class

Drop Precedence	Class 1	Class 2	Class 3	Class 4
Low drop precedence	001010	010010	011010	100010
Medium drop precedence	001100	010100	011100	100100
High drop precedence	001110	010110	011110	100110

Expedited Forwarding PHB

Resource Reservation Protocol (RSVP), a component of the integrated services model, provides a guaranteed bandwidth service. Applications, such as Voice over IP (VoIP), video, and online trading programs, require this type of service. The EF PHB, a key ingredient of DiffServ, supplies this kind of service by providing low loss, low latency, low jitter, and assured bandwidth service.

You can implement EF by using priority queueing (PQ) and rate limiting on the class (or BA). When implemented in a DiffServ network, EF PHB provides a virtual leased line or premium service. For optimal efficiency, however, you should reserve EF PHB for only the most critical applications because, in instances of traffic congestion, it is not feasible to treat all or most traffic as high priority.

EF PHB is suited for applications such as VoIP that require low bandwidth, guaranteed bandwidth, low delay, and low jitter.

IP Precedence for ToS

IP precedence allows you to specify the class of service (CoS) for a packet. You use the three precedence bits in the IPv4 header's type of service (ToS) field for this purpose.

Using the ToS bits, you can define up to six classes of service. Other features configured throughout the network can then use these bits to determine how to treat the packet. These other QoS features can assign appropriate traffic-handling policies including congestion management strategy and bandwidth allocation. For example, although IP precedence is not a queueing method, queueing methods such as weighted fair queueing (WFQ) and Weighted Random Early Detection (WRED) can use the IP precedence setting of the packet to prioritize traffic.

By setting precedence levels on incoming traffic and using them with the ACNS software QoS queueing features, you can create differentiated service. You can use features, such as policy-based routing (PBR) and Committed Access Rate (CAR), to set the precedence based on an extended access list classification. For example, you can assign the precedence based on the application or user or by destination and source subnetwork.

So that each subsequent network element can provide service based on the determined policy, IP precedence is usually deployed as close to the edge of the network or the administrative domain as possible. IP precedence is an edge function that allows core or backbone QoS features, such as WRED, to forward traffic based on CoS. You can also set IP precedence in the host or network client, but this setting can be overridden by the service provisioning policy of the domain within the network.

The following QoS features can use the IP precedence field to determine how traffic is treated:

- Distributed-WRED
- WFQ
- CAR

How the IP Precedence Bits Are Used to Classify Packets

You use the three IP precedence bits in the ToS field of the IP header to specify a CoS assignment for each packet. You can partition traffic into up to six classes—the remaining two classes are reserved for internal network use—and then use policy maps and extended ACLs to define network policies in terms of congestion handling and bandwidth allocation for each class.

Each precedence corresponds to a name. These names, which continue to evolve, are defined in RFC 791. The numbers and their corresponding names, are listed from least to most important.

IP precedence allows you to define your own classification mechanism. For example, you might want to assign the precedence based on an application or an access router. IP precedence bit settings 96 and 112 are reserved for network control information, such as routing updates.

The IP precedence field occupies the three most significant bits of the ToS byte. Only the three IP precedence bits reflect the priority or importance of the packet, not the full value of the ToS byte.

Use the **ip dscp {client | server} {cache-hit | cache-miss} set-tos tos-packets** command to specify either of the two arguments—IP precedence or ToS byte value—to set the same ToS. You may specify either the ToS byte value or IP precedence; one is required. IP precedence uses the three precedence bits in the ToS field of the IPv4 header to specify the class of service for each packet. The ToS byte in the IP header defines the three high-order bits as IP precedence bits and the five low-order bits as ToS bits. The ToS byte value is written to the five low-order bits (bits 0 to 4) of the ToS byte in the IP header of a packet. The IP precedence value is written to the three high-order bits (bits 5 to 7) of the ToS byte in the IP header of a packet.

The following is a list of precedence names:

- **critical**
- **flash**
- **flash-override**
- **immediate**
- **internet**
- **network**
- **priority**
- **routine**

The following is a list of ToS names:

- **max-reliability**
- **max-throughput**
- **min-delay**
- **min-monetary-cost**
- **normal**

Valid values for *tos-packets* are listed in [Table 2-10](#).

Table 2-10 *tos-packets Values*

Value, Precedence, or ToS Name	Description ¹
0–127	Sets the ToS value.
critical	Sets packets with critical precedence (80).

Table 2-10 *tos-packets Values (continued)*

Value, Precedence, or ToS Name	Description ¹
flash	Sets packets with flash precedence (48).
flash-override	Sets packets with flash override precedence (64).
immediate	Sets packets with immediate precedence (32).
internet	Sets packets with internetwork control precedence (96).
max-reliability	Sets packets with maximum reliable ToS (2).
max-throughput	Sets packets with maximum throughput ToS (4).
min-delay	Sets packets with minimum delay ToS (8).
min-monetary-cost	Sets packets with minimum monetary cost ToS (1).
network	Sets packets with network control precedence (112).
normal	Sets packets with normal ToS (0).
priority	Sets packets with priority precedence (16).

1. The number in parentheses denotes the ToS value for each IP precedence or ToS name setting.

Examples

The following example configures a default gateway for the Content Engine:

```
ContentEngine(config) # ip default-gateway 192.168.7.18
```

The following example disables the default gateway:

```
ContentEngine(config) # no ip default-gateway
```

The following example configures a static IP route for the Content Engine:

```
ContentEngine(config) # ip route 172.16.227.128 255.255.255.0 172.16.227.250
```

The following example negates the static IP route:

```
ContentEngine(config) # no ip route 172.16.227.128 255.255.255.0 172.16.227.250
```

The following example configures a default domain name for the Content Engine:

```
ContentEngine(config) # ip domain-name cisco.com
```

The following example negates the default domain name:

```
ContentEngine(config) # no ip domain-name
```

The following example configures a name server for the Content Engine:

```
ContentEngine(config) # ip name-server 10.11.12.13
```

The following example disables the name server:

```
ContentEngine(config) # no ip name-server 10.11.12.13
```

Related Commands

show ip route

ip access-group

To control connections on a specific interface by applying a predefined access list, use the **ip access-group** interface configuration command. To remove access control lists, use the **no** form of this command.

ip access-group {*acl-name* | *acl-num*} {**in** | **out**}

no ip access-group {*acl-name* | *acl-num*} {**in** | **out**}

Syntax Description

<i>acl-name</i>	Alphanumeric identifier of up to 30 characters, beginning with a letter that identifies the ACL to apply to the current interface.
<i>acl-num</i>	Numeric identifier that identifies the access list to apply to the current interface. For standard access lists, the valid range is 1–99; for extended access lists, the valid range is 100–199.
in	Applies the specified access list to inbound packets on the current interface.
out	Applies the specified access list to outbound packets on the current interface.

Defaults

No default behavior or values

Command Modes

interface configuration

Usage Guidelines

Use the **ip access-group** interface configuration command to activate an access list on a particular interface. You can use one outbound access list and one inbound access list on each interface.

Before entering the **ip access-group** command, enter interface configuration mode for the interface to which you want to apply the access list. Define the access list to apply using the **ip access-list** command.

Examples

The following commands apply the access list named *acl-out* to the outbound traffic on the interface FastEthernet 0/0:

```
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# ip access-group acl-out out
```

Related Commands

clear ip access-list counters
ip access-list
show ip access-lists

ip access-list

To create and modify access lists for controlling access to interfaces or applications, use the **ip access-list standard** or **ip access-list extended** global configuration commands. To remove access control lists, use the **no** form of this command.

ip access-list {standard | extended} {acl-name | acl-num}

no ip access-list {standard | extended} {acl-name | acl-num}

Syntax Description

standard	Enables the standard ACL configuration mode.
extended	Enables the extended ACL configuration mode.
<i>acl-name</i>	Access list to which all commands entered from ACL configuration mode apply, using an alphanumeric string of up to 30 characters, beginning with a letter.
<i>acl-num</i>	Access list to which all commands entered from access list configuration mode apply, using a numeric identifier. For standard access lists, the valid range is 1 to 99; for extended access lists, the valid range is 100 to 199.
insert	(Optional) Inserts the conditions following the specified line number into the access list.
<i>line-num</i>	Identifies the entry at a specific line number in the access list.
deny	Causes packets that match the specified conditions to be dropped.
permit	Causes packets that match the specified conditions to be accepted for further processing.
<i>source-ip</i>	Source IP address. The number of the network or host from which the packet is being sent, specified as a 32-bit quantity in 4-part dotted decimal format (for example, 0.0.0.0).
<i>wildcard</i>	(Optional) Portions of the preceding IP address to match, expressed using 4-digit, dotted-decimal notation. Bits to match are identified by a digital value of 0; bits to ignore are identified by a 1. Note For standard IP ACLs, the <i>wildcard</i> argument of the ip access-list command is always optional. If the host keyword is specified for a standard IP ACL, then the <i>wildcard</i> argument is not allowed.
host	Matches the following IP address.
any	Matches any IP address.
delete	Deletes the specified entry.
<i>line-num</i>	Entry at a specific line number in the access list.
list	Lists the specified entries (or all entries when none are specified).
<i>start-line-num</i>	Line number from which the list begins.
<i>end-line-num</i>	(Optional) Last line number in the list.
move	Moves the specified entry in the access list to a new position in the list.
<i>old-line-num</i>	Line number of the entry to move.
<i>new-line-num</i>	New position of the entry. The existing entry is moved to the following position in the access list.
exit	Returns to the CLI global configuration mode prompt.

gre	Matches packets using the Generic Routing Encapsulation protocol.
ip	Matches all IP packets.
<i>proto-num</i>	(Optional) IP protocol number.
tcp	Matches packets using the TCP protocol.
udp	Matches packets using the UDP protocol.
<i>operator</i>	(Optional) Operator to use with specified ports, where lt = less than, gt = greater than, eq = equal to, neq = not equal to, and range = an inclusive range.
<i>port</i>	(Optional) Port, using a number (0–65535) or a keyword; 2 port numbers are required with range . Table 2-11 and Table 2-12 list the UDP and TCP keywords.
<i>dest-ip</i>	Destination IP address. The number of the network or host to which the packet is being sent, specified as a 32-bit quantity in 4-part dotted decimal format (for example, 0.0.0.0).
established	(Optional) Matches TCP packets with the ACK or RST bits set.
icmp	Matches ICMP packets.
<i>icmp-type</i>	(Optional) Matches by ICMP message type (0–255).
<i>code</i>	(Optional) Used with <i>icmp-type</i> to further match by ICMP code type (0–255).
<i>icmp-msg</i>	(Optional) Matches by a combination of ICMP message type and code types, as expressed by the keywords shown in Table 2-13 .

Defaults

An access list drops all packets unless you configure at least one **permit** entry.

Command Modes

global configuration

Usage Guidelines**Standard ACL Configuration Mode Commands**

To work with a standard access list, enter the **ip access-list standard** command from the global configuration mode prompt. The CLI enters a configuration mode in which all subsequent commands apply to the current access list, and the following prompt appears:

```
ContentEngine(config-std-nacl)#
```

To work with entries in a standard access list, enter the following commands from the standard ACL configuration mode (config-std-nacl)#:

To add a line to the standard IP ACL, enter the following command:

For example, choose a purpose (permit or deny) that specifies whether a packet is to be passed or dropped, enter the source IP address, and enter the source IP wildcard address as follows:

```
[insert line-num] {deny | permit} {source-ip [wildcard] | host source-ip | any}
```

To delete a line from the standard IP ACL, enter the following command:

```
delete line-num
```

To display a list of specified entries within the standard IP ACL, enter the following command:

```
list [start-line-num [end-line-num]]
```

To move a line to a new position within the standard IP ACL, enter the following command:

```
move old-line-num new-line-num
```

To return to the CLI global configuration mode prompt, enter the following command:

```
exit
```

To negate a standard IP ACL, enter the following command:

```
no {deny | permit} {source-ip [wildcard] | host source-ip | any}
```

Extended ACL Configuration Mode Commands

To work with an extended access list, enter the **ip access-list extended** command from the global configuration mode prompt. The CLI enters a configuration mode in which all subsequent commands apply to the current access list, and the following prompt appears:

```
ContentEngine(config-ext-nacl)#
```

To work with entries in an extended access list, enter the following commands from the extended ACL configuration mode (config-ext-nacl)#:

To delete a line from the extended IP ACL, enter the following command:

```
delete line-num
```

To move a line to a new position within the extended IP ACL, enter the following command:

```
move old-line-num new-line-num
```

To display a list of specified entries within the standard IP ACL, enter the following command:

```
list [start-line-num [end-line-num]]
```

To return to the CLI global configuration mode prompt, enter the following command:

```
exit
```

To add a condition to the extended IP ACL, note that the options depend on the chosen protocol.

For IP, enter the following command to add a condition:

```
[insert line-num] {deny | permit} {gre | ip | proto-num} {source-ip [wildcard] | host source-ip | any} {dest-ip [wildcard] | host dest-ip | any}
```

```
no {deny | permit} {gre | ip | proto-num} {source-ip [wildcard] | host source-ip | any} {dest-ip [wildcard] | host dest-ip | any}
```

where if you enter *proto-num* is 47 or 0, they represent the equivalent value for GRE or IP.

For TCP, enter the following command to add a condition:

```
[insert line-num] {deny | permit} {tcp | proto-num} {source-ip [wildcard] | host source-ip | any}
[operator port [port]] {dest-ip [wildcard] | host dest-ip | any} [operator port [port]]
[established]

no {deny | permit} {tcp | proto-num} {source-ip [wildcard] | host source-ip | any} [operator port
[port]] {dest-ip [wildcard] | host dest-ip | any} [operator port [port]] [established]
```

where *proto-num* can be 6, which is the equivalent value for TCP.

For UDP, enter the following command to add a condition:

```
[insert line-num] {deny | permit} {udp | proto-num} {source-ip [wildcard] | host source-ip | any}
[operator port [port]] {dest-ip [wildcard] | host dest-ip | any} [operator port [port]]

no {deny | permit} {udp | proto-num} {source-ip [wildcard] | host source-ip | any} [operator port
[port]] {dest-ip [wildcard] | host dest-ip | any} [operator port [port]]
```

where *proto-num* can be 17, which is the equivalent value for UDP.

For ICMP, enter the following command to add a condition:

```
[insert line-num] {deny | permit} {icmp | proto-num} {source-ip [wildcard] | host source-ip | any}
{dest-ip [wildcard] | host dest-ip | any} [icmp-type [code] | icmp-msg]

no {deny | permit} {icmp | proto-num} {source-ip [wildcard] | host source-ip | any} {dest-ip
[wildcard] | host dest-ip | any} [icmp-type [code] | icmp-msg]
```

where *proto-num* can be 2, which is the equivalent value for ICMP.

For extended IP ACLs, the **wildcard** keyword is required if the **host** keyword is not specified. For a list of the keywords that you can use to match specific ICMP message types and codes, see [Table 2-13](#). For a list of supported UDP and TCP keywords, see [Table 2-11](#) and [Table 2-12](#).

Use access lists to control access to specific applications or interfaces on a Content Engine. An access control list consists of one or more condition entries that specify the kind of packets that the Content Engine will drop or accept for further processing. The Content Engine applies each entry in the order in which it occurs in the access list, which by default, is the order in which you configured the entry.

The following are some examples of how IP ACLs can be used in environments that have Content Engines:

- A Content Engine resides on the customer premises and is managed by a service provider, and the service provider wants to secure the device for its management only.
- A Content Engine is deployed anywhere within the enterprise. As with routers and switches, the administrator wants to limit Telnet, SSH, and Content Engine GUI access to the IT source subnets.
- An application layer proxy firewall with a hardened outside interface has no ports exposed. (*Hardened* means that the interface carefully restricts which ports are available for access, primarily for security reasons. With an outside interface, many types of security attacks are possible.) The Content Engine's outside address is Internet global, and its inside address is private. The inside interface has an IP ACL to limit Telnet, SSH, and Content Engine GUI access to the Content Engine.

- A Content Engine is deployed as a reverse proxy in an untrusted environment. The Content Engine administrator wishes to allow only port 80 inbound traffic on the outside interface and outbound connections on the back-end interface.
- A Content Engine using WCCP is positioned between a firewall and an Internet router or a subnet off the Internet router. Both the Content Engine and the router must have IP ACLs.

Within ACL configuration mode, you can use the editing commands (**list**, **delete**, and **move**) to display the current condition entries, to delete a specific entry, or to change the order in which the entries are evaluated. To return to global configuration mode, enter **exit** at the ACL configuration mode prompt.

To create an entry, use a **deny** or **permit** keyword and specify the type of packets that you want the Content Engine to drop or to accept for further processing. By default, an access list denies everything because the list is terminated by an implicit **deny any** entry. You must include at least one **permit** entry to create a valid access list.

After creating an access list, you can include the access list in an access group using the **access-group** command, which determines how the access list is applied. You can also apply the access list to a specific application using the appropriate command. A reference to an access list that does not exist is the equivalent of a **permit any** condition statement.

To work with access lists, enter either the **ip access-list standard** or **ip access-list extended** global configuration command. Identify the new or existing access list with a name up to 30 characters long beginning with a letter or with a number. If you use a number to identify a standard access list, it must be between 1 and 99; for an extended access list, use a number from 100 to 199. You must use a standard access list for providing access to the SNMP server or to the TFTP gateway/server. However, you can use either a standard access list or an extended access list for providing access to the WCCP application.

After you identify the access list, the CLI enters the appropriate configuration mode and all subsequent commands apply to the specified access list.

ip access-list standard Command

You typically use a standard access list to allow connections from a host with a specific IP address or from hosts on a specific network. To allow connections from a specific host, use the **permit host** *source-ip* option and replace *source-ip* with the IP address of the specific host.

To allow connections from a specific network, use the **permit source-ip wildcard** option. Replace *source-ip* with a network ID or the IP address of any host on the network that you want to specify. Replace *wildcard* with the dotted decimal notation for a mask that is the reverse of a subnet mask, where a 0 indicates a position that must be matched and a 1 indicates a position that does not matter. For instance, the wildcard 0.0.0.255 causes the last eight bits in the source IP address to be ignored. Therefore, the **permit 192.168.1.0 0.0.0.255** entry allows access from any host on the 192.168.1.0 network.

ip access-list extended Command

Use an extended access list to control connections based on the destination IP address or based on the protocol type. You can combine these conditions with information about the source IP address to create more restrictive conditions. [Table 2-11](#) lists the UDP keywords that you can use with extended access lists.

Table 2-11 UDP Keywords and Port Numbers

CLI Keyword	Description	UDP Port Number
bootpc	Bootstrap Protocol (BOOTP) client service	68
bootps	Bootstrap Protocol (BOOTP) server service	67
domain	Domain Name System (DNS) service	53
netbios-dgm	NetBIOS datagram service	138
netbios-ns	NetBIOS name resolution service	137
netbios-ss	NetBIOS session service	139
nfs	Network File System service	2049
ntp	Network Time Protocol settings	123
snmp	Simple Network Management Protocol service	161
snmptrap	SNMP traps	162
tacacs	Terminal Access Controller Access Control System remote authentication service	49
tftp	Trivial File Transfer Protocol service	69
wccp	Web Cache Communication Protocol service	2048

Table 2-12 lists the TCP keywords that you can use with extended access lists.

Table 2-12 TCP Keywords and Port Numbers

CLI Keyword	Description	TCP Port Number
domain	Domain Name System	53
exec	Remote process execution	512
ftp	File Transfer Protocol service	21
ftp-data	FTP data connections (used infrequently)	20
https	Secure HTTP service	443
nfs	Network File System service applications	2049
rtsp	Real-Time Streaming Protocol applications	554
ssh	Secure Shell login	22
tacacs	Terminal Access Controller Access Control System authentication service	49
telnet	Remote login using telnet	23
www	World Wide Web (HTTP) service	80

Table 2-13 lists the keywords that you can use to match specific ICMP message types and codes.

Table 2-13 **Keywords for ICMP Message Type and Code**

Field	Description
administratively-prohibited	Messages that are administratively prohibited from being allowed access.
alternate-address	Messages that specify alternate IP addresses.
conversion-error	Messages that denote a datagram conversion error.
dod-host-prohibited	Messages that signify a Department of Defense (DoD) protocol Internet host denial.
dod-net-prohibited	Messages that specify a DoD protocol network denial.
echo	Messages that are used to send echo packets to test basic network connectivity.
echo-reply	Messages that are used to send echo reply packets.
general-parameter-problem	Messages that report general parameter problems.
host-isolated	Messages that indicate that the host is isolated.
host-precedence-unreachable	Messages that have been received with the protocol field of the IP header set to 1 (ICMP) and the type field in the ICMP header set to 3 (Host Unreachable). This is the most common response. Large numbers of this datagram type on the network are indicative of network difficulties or may be indicative of hostile actions.
host-redirect	Messages that specify redirection to a host.
host-tos-redirect	Messages that specify redirection to a host for type of service-based (ToS) routing.
host-tos-unreachable	Messages that denote that the host is unreachable for ToS-based routing.
host-unknown	Messages that specify that the host or source is unknown.
host-unreachable	Messages that specify that the host is unreachable.
information-reply	Messages that contain domain name replies.
information-request	Messages that contain domain name requests.
mask-reply	Messages that contain subnet mask replies.
mask-request	Messages that contain subnet mask requests.
mobile-redirect	Messages that specify redirection to a mobile host.
net-redirect	Messages that are used for redirection to a different network.
net-tos-redirect	Messages that are used for redirection to a different network for ToS-based routing.
net-tos-unreachable	Messages that specify that the network is unreachable for the ToS-based routing.
net-unreachable	Messages that specify that the network is unreachable.
network-unknown	Messages that denote that the network is unknown.
no-room-for-option	Messages that specify the requirement of a parameter, but that no room is unavailable for it.

Table 2-13 **Keywords for ICMP Message Type and Code (continued)**

Field	Description
option-missing	Messages that specify the requirement of a parameter, but that parameter is not available.
packet-too-big	Messages that specify that the ICMP packet requires fragmentation but the Do Not Fragment (DF) bit is set.
parameter-problem	Messages that signify parameter-related problems.
port-unreachable	Messages that specify that the port is unreachable.
precedence-unreachable	Messages that specify that host precedence is not available.
protocol-unreachable	Messages that specify that the protocol is unreachable.
reassembly-timeout	Messages that specify a timeout during reassembling of packets.
redirect	Messages that have been received with the protocol field of the IP header set to 1 (ICMP) and the type field in the ICMP header set to 5 (Redirect). ICMP redirect messages are used by routers to notify the hosts on the data link that a better route is available for a particular destination.
router-advertisement	Messages that contain ICMP router discovery messages called router advertisements.
router-solicitation	Messages that are multicast to ask for immediate updates on neighboring router interface states.
source-quench	Messages that have been received with the protocol field of the IP header set to 1 (ICMP) and the type field in the ICMP header set to 4 (Source Quench). This datagram may be used in network management to provide congestion control. A source quench packet will be issued when a router is beginning to lose packets due to the transmission rate of a source. The source quench is a request to the source to reduce the rate of a datagram transmission.
source-route-failed	Messages that specify the failure of a source route.
time-exceeded	Messages that specify information about all instances when specified times were exceeded.
timestamp-reply	Messages that contain time-stamp replies.
timestamp-request	Messages that contain time-stamp requests.
traceroute	Messages that specify the entire route to a network host from the source.
ttl-exceeded	Messages that specify that ICMP packets have exceeded the Time-To-Live configuration.
unreachable	Messages that are sent when packets are denied by an access list; these packets are not dropped in the hardware but generate the ICMP-unreachable message.

Examples

The following example shows how to create an access list to allow all web traffic and to only allow a specific host administrative access using Secure Shell (SSH):

```
ContentEngine(config)# ip access-list extended example
ContentEngine(config-ext-nacl)# permit tcp any any eq www
ContentEngine(config-ext-nacl)# permit tcp host 10.1.1.5 any eq ssh
ContentEngine(config-ext-nacl)# exit
```

The following example shows how to activate the access list for an interface:

```
ContentEngine(config)# interface gigabitethernet 1/0
ContentEngine(config-if)# ip access-group example in
ContentEngine(config-if)# exit
```

The following example shows how this configuration appears when you enter the **show running-configuration** command:

```
...
!
interface GigabitEthernet 1/0
 ip address 10.1.1.50 255.255.0.0
 ip access-group example in
 exit
...
ip access-list extended example
 permit tcp any any eq www
 permit tcp host 10.1.1.5 any eq ssh
 exit
...

```

Related Commands

clear ip access-list counters
ip access-group
show ip access-list

ip address

To configure the IP address or subnet mask, or to negotiate an IP address from DHCP on the Content Engine interface, use the **ip address** interface configuration command. To disable this function, use the **no** form of this command.

ip address {*ip-address ip-subnet* [*secondary*] | **dhcp** [*client-id id* [*hostname name*] | **hostname name** [*client-id id*]}

no ip address {*ip-address ip-subnet* [*secondary*] | **dhcp** [*client-id id* [*hostname name*] | **hostname name** [*client-id id*]}

Syntax Description

<i>ip-address</i>	IP address.
<i>ip-subnet</i>	IP subnet mask.
secondary	(Optional) Makes this IP address a secondary address.
dhcp	Sets the IP address negotiated over DHCP.
client-id	(Optional) Specifies the client identifier.
<i>id</i>	Client identifier.
hostname	(Optional) Specifies the hostname.
<i>name</i>	Hostname.

Defaults

No default behavior or values

Command Modes

interface configuration

Usage Guidelines

Use this command to set or change the IP address, subnet mask, or DHCP IP address negotiation of the Content Engine network interfaces. The change in the IP address takes place immediately.

The **ip address** interface configuration command allows the configuration of secondary IP addresses for a specified interface as follows:

```
ContentEngine(config-if)# ip address ip_address netmask [secondary]
```

You can specify up to four secondary IP addresses for each interface. The same IP address cannot be assigned to more than one interface. The secondary IP address becomes active only after a primary IP address is configured. The secondary IP addresses are disabled when the interface is shut down and are enabled when the interface is brought up.

Use the **no** form of the command to disable a specific IP address.



Note

No two interfaces can have IP addresses in the same subnet.

Use the **ip address dhcp** command to negotiate a reusable IP address from DHCP.

Examples

The following example shows how to configure the port-channel interface with an IP address of 10.10.10.10 and a netmask of 255.0.0.0:

```
ContentEngine# configure
ContentEngine(config)# interface PortChannel 2
ContentEngine(config-if)# ip address 10.10.10.10 255.0.0.0
```

The following example deletes the IP address configured on the interface:

```
ContentEngine(config-if)# no ip address
```

The following example enables an interface for DHCP:

```
ContentEngine(config-if)# ip address dhcp
```

The following example configures a client identifier and hostname on the Content Engine to be sent to the DHCP server:

```
ContentEngine(config-if)# ip address dhcp client-id myclient hostname myhost
```

The following example configures the primary IP address:

```
ContentEngine(config-if)# ip address ip_address netmask
```

Related Commands

```
interface
show interface
show running-config
show startup-config
```

iptv program-manager

To configure the IP/TV Program Manager database and files, use the **iptv program-manager EXEC** command.

iptv program-manager {database {backup | clear | dump journal | restore} | start | stop}

Syntax Description

database	Configures the IP/TV Program Manager database commands.
backup	Backs up the IP/TV Program Manager database to a file.
clear	Clears the IP/TV Program Manager database.
dump	Configures dumping of the IP/TV Program Manager database.
journal	Dumps the transaction journal in the IP/TV Program Manager database to a file.
restore	Restores the IP/TV Program Manager database from a file.
start	Starts the IP/TV Program Manager functions.
stop	Stops the IP/TV Program Manager functions.

Defaults

No default behavior or values

Command Modes

EXEC

Usage Guidelines

IP/TV Program Manager is a device mode of the Content Engine. Use the **iptv program-manager EXEC** command to configure IP/TV Program Manager and manage the IP/TV Program Manager database.



Note

The **iptv program-manager EXEC** command is available only when the Content Engine is running as IP/TV Program Manager.

Starting and Stopping IP/TV Program Manager

You can start or stop IP/TV Program Manager using the **iptv program-manager EXEC** command.

Use the **device mode program-manager** command to change the device mode to IP/TV Program Manager.

Use the **iptv program-manager start** command to start the IP/TV Program Manager.

Use the **iptv program-manager stop** command to stop the IP/TV Program Manager.

Managing the Database in IP/TV Program Manager

You can manage the IP/TV Program Manager database using the **iptv program-manager database EXEC** commands.

Use the **iptv program-manager database backup** command to back up your IP/TV Program Manager database.

Use the **iptv program manager database clear** command to clear your IP/TV Program Manager database.

Use the **iptv program-manager database dump** command to transfer the data from the IP/TV Program Manager database to another file.

Use the **iptv program-manager database dump journal** command to access the IP/TV Program Manager journaling database.

Use the **iptv program-manager restore** command to restore the IP/TV Program Manager database from the backup file.

Examples

The following example shows how to start IP/TV Program Manager:

```
iptv-dev-565-1# iptv program-manager start
Starting IP/TV Program Manager services
Program Manager services successfully started
iptv-dev-565-1#
```

Related Commands

debug iptv program-manager

kernel kdb

To enable access to the kernel debugger (kdb), use the **kernel kdb** global configuration command. Once enabled, kdb is automatically activated if kernel problems occur, or you can manually activate it from the local console for the ACNS device by pressing the required key sequence. To disable the kernel debugger, use the **no** form of this command.

kernel kdb

no kernel kdb

Syntax Description

This command has no arguments or keywords.

Defaults

In the ACNS software, Release 5.0.3 and later releases, kdb is disabled by default.

In the ACNS software earlier than Release 5.0.3, kdb is enabled by default.

Command Modes

global configuration

Usage Guidelines

Once enabled, kdb is automatically activated when kernel problems occur. Once activated, all normal functioning of the ACNS device is suspended until kdb is manually deactivated. The kdb prompt looks like this prompt:

```
[0]kdb>
```

To deactivate kdb, enter **go** at the kdb prompt. If kdb was automatically activated because of kernel problems, the system generates a core dump and restarts. If you activated kdb manually for diagnostic purposes, the system resumes normal functioning in whatever state it was when you activated kdb. In either case, if you enter **reboot**, the system restarts and normal operation resumes.

With the ACNS software earlier than Release 5.0.3, kdb is enabled by default and cannot be disabled. To activate it manually from the local console, press **Ctrl-_** (the **Ctrl** key and the **Underscore** key pressed simultaneously).

In the ACNS software, Release 5.0.3 and later releases, kdb is disabled by default and you must enter the **kernel kdb** command in global configuration mode to enable it. If kdb has been previously enabled, you can enter the **no kernel kdb** global configuration command to disable it. When kdb is enabled, you can activate it manually from the local console by pressing **Ctrl-_** followed by **Ctrl-B**.

Examples

The following example shows how to enable kdb:

```
ContentEngine(config)# kernel kdb
```

The following example shows how to disable kdb:

```
ContentEngine(config)# no kernel kdb
```

ldap

To configure the Content Engine to perform user authentication with a Lightweight Directory Access Protocol (LDAP) server, use the **ldap** global configuration command. To disable **ldap** options, use the **no** form of this command.

```
ldap server { administrative-dn name | administrative-passwd passwd | allow-mode | base
baseword | enable | filter filterword | group { active-directory enable | custom uid enable |
organizationUnit enable | static { enable | group-attribute gid | member-attribute
{ custom-member memid | member | uniquemember } | nested { enable | level 1-100 } } | host
{ hostname | hostipaddress } [ primary | secondary ] | password-expiry { enable | redirect-url
url } | policy-redirect { append-request-url | attribute name | enable | redirect-url url |
version-number num } | port port-num | retransmit retries | timeout seconds |
userid-attribute useridword | version ver_num }
```

```
no ldap server { administrative-dn | administrative-passwd | allow-mode | base | enable | filter |
group { active-directory enable | custom uid enable | organizationUnit enable | static
{ enable | group-attribute gid | member-attribute { custom-member memid | member |
uniquemember } | nested { enable | level 1-100 } } | host { hostname | hostipaddress }
[ primary | secondary ] | password-expiry { enable | redirect-url url } | policy-redirect
{ append-request-url | attribute name | enable | redirect-url url | version-number num } |
port | retransmit | timeout | userid-attribute | version }
```

Syntax Description

server	Configures LDAP server parameters.
enable	Enables the LDAP feature.
administrative-dn	Sets the administrative distinguished name.
<i>name</i>	Administrative distinguished name.
administrative-passwd	Sets the administrative password.
<i>passwd</i>	Administrative password.
allow-mode	Allows access to users when the LDAP server is unavailable.
base	Sets the base distinguished name of the starting point for the search in the LDAP database.
<i>baseword</i>	Base value. There is no default.
enable	Enables the HTTP request authentication with the LDAP server.
filter	Sets the LDAP filter for the authentication group.
<i>filterword</i>	Text for the LDAP filter. There is no default.
group	Configures LDAP groups.
active-directory	Obtains the group name from the <i>memberOf</i> attribute of the user account.
enable	Enables the Active Directory group membership.
custom	Obtains the group name from the <i>custom</i> attribute of the user account.
<i>uid</i>	Custom attribute name of the user account.
enable	Enables the customized attribute name group membership.
organizationUnit	Obtains the group name from the <i>ou</i> attribute of the user account.
enable	Enables the organizationUnit group membership.
static	Configures the LDAP static groups.

enable	Enables the static group query for the user membership.
group-attribute	Specifies the name of the static group attribute.
<i>gid</i>	Static group attribute name (for example, cn, gid).
member-attribute	Specifies the name of the group member attribute to query.
custom-member	Configures the custom static group member attribute name.
<i>memid</i>	Custom static group member attribute name.
member	Configures the static group member attribute name.
uniquemember	Specifies the static group member attribute name.
nested	Obtains names of nested static groups.
enable	Enables the nested static group query for the user membership.
level	Specifies the level of nested static group to query.
<i>1-100</i>	Nested level of static group (the default is 1).
host	Sets host parameters.
<i>hostname</i>	Hostname of the LDAP server. Up to two servers can be named.
<i>hostipaddress</i>	IP address of the LDAP server.
primary	(Optional) Specifies the host as the primary host.
secondary	(Optional) Specifies the host as the secondary host.
password-expiry	Configures the authorization password expiration.
enable	Enables the support for the authorization password expiration.
redirect-url	Defines the URL that allows a user to reset an expired password.
<i>url</i>	URL to redirect to if the user password has expired.
policy-redirect	Configures the policy redirection support.
append-request-url	Enables the Content Engine to append the request URL (original destination URL) to the LDAP AUP redirect URL.
attribute	Defines the LDAP attribute name that determines if a user has accepted the site policy.
<i>name</i>	LDAP attribute name under which the policy version value is stored.
enable	Enables the policy redirection support.
redirect-url	Defines the URL to redirect to for acceptable usage policy acknowledgement.
<i>url</i>	URL to redirect to if the user still has accepted the usage policy.
version-number	Defines the current policy version number.
<i>num</i>	Policy version number (1–99999999).
port	Sets the TCP port for the LDAP authentication server.
<i>port-num</i>	LDAP server port number (1–65535). The default is 389.
retransmit	Specifies the number of transmission attempts to an active server.
<i>retries</i>	Number of transmission attempts for a transaction (1–3). The default is 2.
timeout	Sets the time to wait for an LDAP server to reply.
<i>seconds</i>	Waiting time in seconds (1–20). The default is 5 seconds; the minimum is 1 second and the maximum is 20 seconds.
userid-attribute	Sets the user ID attribute on the LDAP server.
<i>useridword</i>	Value for the user ID attribute. The default is uid.

version	Sets the LDAP version number.
<i>ver_num</i>	LDAP version number (2–3). The default is 2.

Defaults

nested level: 1
version *ver_num*: 2
allow-mode: enabled
port *port-num.* 389
timeout *seconds*: 5
useridword: uid

Command Modes

global configuration

Usage Guidelines

LDAP is a directory service protocol that is simpler than the TCP/IP-based version of Directory Access Protocol (DAP) and can be used to access information directories.

A Content Engine can be configured to restrict user Internet access by using an LDAP server for authentication purposes, which provides most of the services of the X.500 protocol with less complexity and overhead.

**Note**

To exclude domains from LDAP authentication, use the **rule no-auth domain** global configuration command. Authentication challenges from LDAP, RADIUS, TACACS+, or SSH take place only if the request does not match the specified **no-auth** pattern.

Typically, an LDAP client (the Content Engine) queries an LDAP server and obtains the user's credentials, such as the user's account expiration, privileges, and group membership, from the remote LDAP directory on an OpenLDAP or third-party LDAP server. In the ACNS software, Release 5.1 and later releases, the Content Engine can also authenticate and authorize a user who is configured in a remote Active Directory user database on a Microsoft Active Directory server.

The Content Engine (an LDAP client) works with any of the following types of servers to perform LDAP authentication of HTTP requests:

- OpenLDAP servers (shareware servers)
- Third-party LDAP servers (for example, a Sun Microsystems iPlanet server)
- Microsoft Active Directory (AD) servers

**Note**

The ACNS 5.x software supports LDAP Version 2 and Version 3 and supports all LDAP features except for Secure Authentication and Security Layer (SASL). The Content Engine uses simple (nonencrypted) authentication to communicate with the LDAP server. Future expansion may allow for more security options based on Secure Sockets Layer (SSL), SASL, or certificate-based authentication.

The Active Directory group attribute is an LDAP Version 3 extension. The Content Engine must use LDAP Version 3 to query a Microsoft Active Directory server separately for this information.

**Note**

For features supported on different types of LDAP servers and for guidelines to follow when configuring LDAP authentication of HTTP requests, see Chapter 10 of the *Cisco ACNS Software Configuration Guide for Locally Managed Deployments*.

**Note**

If you do not configure LDAP servers, the LDAP authentication will not be performed.

Retrieve group names from the LDAP database for group-based authorization by entering the **ldap server group** global configuration command. Use this command to instruct the LDAP server, which is performing the database query, to retrieve the names of the groups that the authenticated user belongs to (for example, Marketing or Engineering). The Content Engine uses these retrieved group names to perform LDAP group-based authorization by checking its access lists to determine whether the groups should be granted or denied access to the requested content.

**Note**

The ACNS software, Release 5.1 and later releases support LDAP Version 2 and Version 3 and support all LDAP features except for SASL.

Table 2-14 lists the Content Engine's default configuration for LDAP authentication of HTTP requests.

Table 2-14 Default Configuration for LDAP Authentication of HTTP Requests

Feature or Setting	Default Value
LDAP authentication	Disabled
Allow mode	Disabled
Base distinguished name	None specified (an empty string)
Filter for database searches	None specified
LDAP retransmit attempts	2 times
LDAP server timeout	5 seconds
User ID attribute	uid
Group attribute	
• Organization unit (ou)	Enabled
• Custom attribute	Disabled
• Active Directory (memberOf)	Disabled

Table 2-14 **Default Configuration for LDAP Authentication of HTTP Requests (continued)**

Feature or Setting	Default Value
Static group database queries	Disabled
• Group attribute	None specified
• Group member	None specified
• Nested groups	Disabled
• Nested level	1 level
Administrative distinguished name	None specified
Administrative password	None specified
LDAP version	LDAP Version 2
LDAP port	Port 389
Policy redirect feature	Disabled
Password expiry feature	Disabled
Primary LDAP server	None specified
Secondary LDAP server	None specified

You can also configure the Content Engine to perform group-based authorization after a user has been successfully authenticated. If the Content Engine is configured for group-based authorization, then the Content Engine checks its access lists to determine whether the groups that the authenticated user belongs to should be granted or denied access to the requested content. Based on the results of this LDAP group-based authorization, the Content Engine grants or denies user access to the requested content.

In order for LDAP group-based authorization to occur, you must have completed the following prerequisite tasks on the Content Engine:

- Enabled and configured group name-based access lists, as described in the [“LDAP Group Authentication with Access Lists”](#) section on page 2-296.
- Configured the Content Engine to request that the LDAP server retrieve the names of the groups that the authenticated user belongs to (for example, Marketing or Engineering). You must understand the structure of the user database being queried in order to configure the Content Engine to request this information retrieval. For more information, see the next section, [Understanding the Structure of the LDAP Database](#).

Understanding the Structure of the LDAP Database

In order to configure the LDAP client on the Content Engine to query a remote user database, you must understand the structure of the user database being queried.

LDAP database entries are arranged in a hierarchical directory tree that reflects logical or geographic boundaries. The LDAP directory has the following structure:

- The topmost node of the LDAP directory tree is named as root.
- Below the root, there are organization nodes (o) for companies, states, or national organizations (for example, o=cisco, o=texas, and o=redcross).
- Below organization nodes, there are LDAP group nodes for such organizational units as departments (for example, ou=Marketing and ou=Engineering) and branch offices.
- At the bottom of the tree, there are individual nodes (cn) for common names of people (for example, cn=Jane Smith), documents, and such shared resources as printers.

**Note**

Because the groups named Hardware Engineers and Software Developers are nested under the parent group named Engineering, they are called nested groups. Nested groups allow the LDAP server administrator to create hierarchical relationships that can be used to define inherited group membership.

An LDAP directory can contain such information as text, photographs (JPEGs), URLs, binary data, and public key certificates.

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

The ACNS software, Release 5.1 and later releases support LDAP Version 2 and Version 3 and support all LDAP features except for the SASL.

**Note**

HTTP authentication featuring RADIUS and LDAP existed in the Cache software 2.x releases and was configured through the **radius-server** and **ldap** commands, respectively. For the ACNS 5.x software, the **radius-server authtimeout** option and the **ldap authcache max-entries** and **ldap authcache auth-timeout** options have been removed and are now configurable through the **http authentication cache max-entries** and **timeout** commands, respectively. The **ldap client auth-header** option has been removed and is now configurable through the **http authentication header** command. The **multi-user-prompt** has been removed and replaced by the **http avoid-multiple-user-prompts** option. In addition, the **radius-server** command option **exclude** has been removed. The **rule no-auth domain** command replaces the **radius-server exclude** command; however, there is no replacement available for the **multi-user-prompt** option. The **ldap server** command has the **enable** and **version** options.

Understanding LDAP Active Directory Configuration

An Active Directory database is a user database of a Windows 2000 server. This database can be queried for authentication purposes by LDAP protocols. Typically, a Content Engine LDAP client queries an LDAP server's user database and obtains the user's credentials such as the user's account expiration, privileges, and group membership. In the ACNS 5.x software, the Content Engine LDAP client is also allowed to authenticate and authorize a user configured in a remote Active Directory in a Windows 2000 server database.

Use the **ldap server group active-directory enable** command to activate this feature.

LDAP Group Authentication with Access Lists

ACNS releases prior to 5.1 supported group name-based access lists but did not support static groups. To ensure interoperability of the Content Engine group authentication support with the Microsoft Active Directory database, the ACNS software, Release 5.1 and later releases support LDAP group-based authorization for static groups.

To configure group authentication and authorization using an LDAP server and access control lists, follow these steps. In this scenario, LDAP group access is allowed to the Engineering, Marketing, and Sales groups in company XYZ. The following parameters are configured in the LDAP server for user John Smith from the Engineering group:

```
# John Smith
dn: cn=John Smith,ou=Engineering,dc=xyz,dc=com
cn: John Smith
sn: Smith
telephoneNumber: 1 408 123 4567
uid: jsmith
userpassword: john123
```


Acceptable Use Policy

The ACNS software, Release 5.1 and later releases, support the LDAP acceptable use policy feature. When a user opens a browser session, the Content Engine queries a specific LDAP attribute to determine whether the user has viewed and accepted the acceptable use policy (AUP). If a user has not accepted this policy, then the Content Engine redirects the user to an internal web page with the AUP, which the user must read and accept before being allowed to browse content through the Content Engine. Once the user accepts the policy, the Content Engine sets an LDAP attribute that allows the user full access to browse through the proxy server (the Content Engine).

This LDAP attribute is configurable and is an integer stored in the user's database that represents the version of the policy that the user has accepted. This value is compared against the current version set in the Content Engine. If these values are equal, the user is given access to browse; otherwise, the user is redirected to the configured URL that sends the user to the internal web page that allows the user to read and accept the AUP.

Use the **ldap server policy-redirect enable** global configuration command to enable the AUP. Use the **ldap server policy-redirect redirect-url url** global configuration command to specify the URL to which the user is redirected to view and accept the policy. Use the **ldap server policy-redirect attribute attribute** to define the LDAP attribute that is to be queried for the version that the user has accepted.

LDAP Password Expiration

The LDAP password feature allows you to configure the Content Engine to redirect users to a web page, which will prompt them for a username and password. To configure the LDAP authorization password expiration feature, use the **ldap server password-expiry** global configuration command.

Security Options

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

Excluding Domains

To exclude domains from LDAP authentication, use the **rule no-auth domain** command. Authentication challenges from LDAP, RADIUS, TACACS+, or SSH take place only if the request does not match the specified **no-auth** pattern.

LDAP and RADIUS Considerations

LDAP authentication can be used with Websense and N2H2 URL filtering, but not with RADIUS authentication. Both LDAP and RADIUS rely on different servers, which may require different user IDs and passwords, making RADIUS and LDAP authentication schemes mutually exclusive. If both RADIUS and LDAP are configured on the Content Engine at the same time, LDAP authentication is executed, but RADIUS authentication is not executed.

Hierarchical Caching in Proxy Mode

In some cases, users are located at branch offices. A Content Engine (CE1) can reside with them in the branch office. Another Content Engine (CE2) can reside upstream, with an LDAP server available to both Content Engines for user authentication.



Note

You can configure the **http append proxy-auth-header** global configuration command on the downstream Content Engines to ensure that proxy-authorization information, required by upstream Content Engines, is not stripped from the HTTP request by the downstream Content Engines. You can configure a maximum of 16 upstream IP addresses on each downstream Content Engine.

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

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

If authentication is successful and a cache miss occurs, the request and the credentials are forwarded to CE2. CE2 also contacts the LDAP server to authenticate user 2. CE2 either serves the request out of its cache or forwards the request to the origin server.

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

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

Hierarchical Caching in Transparent Mode

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

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

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

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

In a topology with two Content Engines, assume a scenario where CE1 is operating in transparent mode and CE2 is operating in proxy mode, with the browsers of all users pointing to CE2 as a proxy. Because the browsers are set up to send requests to a proxy, an HTTP 407 message is sent from CE1 back to each user to prompt for credentials. By using the 407 message, the problem of authenticating based on source IP address is avoided. The username and password can be used instead.

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

Server Redundancy

A second LDAP server can be specified with the **host** command option. The additional server provides authentication redundancy and improved throughput, especially when Content Engine load-balancing schemes distribute the requests evenly between the two servers. If the Content Engine cannot connect to either of the two LDAP servers, no authentication takes place, and users who have not been previously authenticated are denied access.

Authentication Cache Size Adjustments

If the authentication cache is not large enough to accommodate all authenticated users at the same time, the Content Engine purges older entries that have not timed out.

Transaction Logging

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

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

Examples

The following example specifies an LDAP server with IP address 10.1.1.1 on port 88 and excludes the domain name mydomain.net from LDAP authentication with the **rule** global configuration command:

```
ContentEngine(config)# ldap server enable
ContentEngine(config)# ldap server host 10.1.1.1 port 88
ContentEngine# show ldap
LDAP parameters:
    State:          Enabled
    Base DN:        <none>
    Filter:         <none>
    Timeout:        5 seconds
    UID Attribute:  uid
    Primary:        10.1.1.1
    Secondary:      <none>
    LDAP port:      88
    Administrative DN: <none>
    Administrative Password: <none>
    LDAP version: 2
ContentEngine(config)# rule enable
ContentEngine(config)# rule no-auth domain mydomain.net

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

The following example shows how to delete an LDAP server:

```
ContentEngine(config)# no ldap server host 10.1.1.1 port 88
```

The following example shows how to specify two LDAP servers as primary and secondary HTTP request authentication servers:

```
ContentEngine(configure)# ldap server host 172.16.1.1 primary
ContentEngine(configure)# ldap server host 172.16.1.2 secondary
```

Related Commands

```
debug authentication http-request
rule
show ldap
show rule
```