



# Configuring Content Preloading and URL Filtering on Standalone Content Engines

---

This chapter provides an overview of content preloading and the different types of URL filtering that are supported with standalone Content Engines that are running the ACNS 5.4.x software or later releases, and describes how to configure content preloading and URL filtering on standalone Content Engines.

This chapter contains the following sections:

- [Configuring Content Preloading for Standalone Content Engines, page 11-2](#)
- [Configuring URL Filtering on Standalone Content Engines, page 11-11](#)
- [Configuring Content Engines to Bypass URL Filtering for Specific HTTP and HTTPS Requests, page 11-43](#)
- [Displaying the Current URL Filtering Configurations, page 11-43](#)
- [Displaying URL Filtering Statistics, page 11-43](#)
- [Clearing URL Filtering Statistics, page 11-44](#)

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



## Note

---

For complete syntax and usage information for the CLI commands used in this chapter, see the *Cisco ACNS Software Command Reference, Release 5.5* publication.

For information about how to configure URL filtering on Content Engines that are registered with a Content Distribution Manager (as opposed to standalone Content Engines), see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*.

---

# Configuring Content Preloading for Standalone Content Engines

This section provides an overview of content preloading for standalone Content Engines that are running the ACNS 5.4.1 software and later releases. It also provides instructions on how to configure this feature on a standalone Content Engine.

Preloaded content is content that is retrieved and stored on a standalone Content Engine because the administrator of that Content Engine scheduled a retrieval of specific content in anticipation of user requests for that content. Content preloading is done by configuring the standalone Content Engine to create a cache request for all the content located at the origin web server that stores the primary content.

You can specify bandwidth limits for the preload process to ensure that bandwidth consumption does not exceed the specified bandwidth limits during the preload process. During the preload process, the Content Engine scans websites several link levels down for content, retrieves the specified content, and stores it locally for future requests. At a specified time, the Content Engine scans several levels of websites to verify that its content is still current, and it updates any content that has changed.



## Note

The ACNS 5.x software can read a file of URLs and preload the specified URL content on the standalone Content Engine. The following type of content can be preloaded on a standalone Content Engine: HTTP URLs and FTP-over-HTTP URLs. All configured HTTP and FTP-over-HTTP parameters and rules apply to the preloaded objects.

HTTP URLs are parsed for username and password, but FTP URLs are not. By design, ACNS software supports authentication for HTTP requests only. We recommend that you use anonymous FTP URLs or use an HTTP server for preloading content. Only anonymous FTP URLs may be preloaded.

## Support of Preloading of NTLM Authenticated Objects

In the ACNS 5.1.1 software and later releases, the preloading of NTLM authenticated objects is supported. This feature allows NTLM authenticated objects (authenticated objects that reside on the servers that authenticate NTLM only) to be preloaded on a Content Engine.



## Note

In the ACNS 5.1.1 software and later releases, NTLMv1 support of preloading NTLM authenticated objects on standalone Content Engines is available. In the ACNS 5.4.1 software release, NTLMv2 support of preloading NTLM authenticated objects was added. The **ntlm version** global configuration command, which was added in the ACNS 5.4.1 software release, has two command options: the **version 1** option to re-enable NTLMv1, and the **version 2** option to enable NTLMv2 on a standalone Content Engine. The **version 1** command option is the default option.

An entry in a URL list file has the following format:

```
URL [depth] [domain-name:host-name:host-domain-name]
```

*hostname* and *host-domain-name* can be null; however, the *domain name* is required if NTLM credentials have been configured. (The separator is required.)

```
http://www.cisco.com 3 apac::
```

If NTLM-related information is not present in the preload URL list file entry, the authentication scheme falls back to basic authentication.

By default, the Content Engine does not cache basic and NTLM authenticated objects. To enable a standalone Content Engine to fetch specific objects and cache these objects that are authenticated with any authentication scheme (basic authentication or NTLM authentication), enter the **http cache-authenticated all** global configuration command.

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

To configure the Content Engine to cache only NTLM authenticated objects, enter the **http cache-authenticated ntlm** global configuration command. The cached objects are tagged as NTLM protected so that subsequent requests for these same objects are subjected to authentication before the Content Engine can serve the content to the client.

Before you preload WMT streaming media files on the Content Engine, you must enable the WMT feature on your Content Engine. If you used the Setup utility to configure WMT caching (as described in [Step 15](#) of the “[Using the Setup Utility to Configure a Basic Configuration on a Standalone Content Engine](#)”) on your Content Engine, then WMT is already enabled on the Content Engine. Otherwise, see the “[Configuring WMT RTSP Streaming and Caching Services on Standalone Content Engines](#)” section on page 9-14 for instructions on how to use the Content Engine CLI (instead of the Setup utility) to enable Windows Media services on a standalone Content Engine before you enable the preloading of Windows Media streaming files for this Content Engine.

## Creating a Preload URL List File

The preload URL list file lists the URLs (HTTP or FTP-over-HTTP) to be preloaded on the Content Engine. This file is maintained by the administrator and must be created on a remote system. This file can be transferred to the standalone Content Engine for preloading access, or accessed from the remote server. The **pre-load url-list-file path** global configuration command specifies the path of this file.



### Note

In the **pre-load url-list-file path** global configuration command, the value for *path* can be a URL or a local file path.

You can place the list of URLs in a file on a local disk. You can also use the **mkdir EXEC** command to make a subdirectory that contains the preload URL list file. For instance, the **mkdir /local1/preload-directory** command creates a subdirectory called preload-directory on local disk /local1.

Each URL in the preload URL list file has an optional depth parameter. The depth parameter specifies how many levels down the preloading will be performed. For example, `http://www.espn.com 3` means download `http://www.espn.com` and all content three levels deep. If the depth level is not specified, then the preload depth level default of 3 is used. The URLs are delimited with a carriage return as follows:

```
<cr>
. . .
http://www.cnn.com 3 <cr>
ftp://ftp.lehigh.edu/ 2 <cr>
http://www.yahoo.com <cr>
. . .
<cr>
```

If you want to preload authenticated content to a Content Engine, the URL list file entry must be written as follows:

```
http://username:password@www.authenticationsite.com/ depth level
```

**Note**

In the ACNS 5.4.1 software and later releases, support for proxy authentication for preloaded content was added. For more information on this topic, see the [“Proxy Authentication Support for Content Preloading”](#) section on page 11-8.

When configuring a preload URL list file through the Content Engine CLI, the **pre-load url-list-file** global configuration command only had the HTTP or FTP option in the ACNS 5.1.x software earlier than the ACNS 5.1.5 software release. There was no method in place to fetch the preload URL list file securely.

In the ACNS 5.1.5 software release, the ability to fetch the preload URL file over HTTPS was added. If a preload URL list file contains usernames and passwords, organizations are now able to fetch the preload URL list file over HTTPS. The actual preloading of HTTPS links is not supported; only the downloading of the preload URL list file through the HTTPS protocol.

## Enabling and Configuring Content Preloading on Standalone Content Engines

You can enable and configure content preloading on a standalone Content Engine from either the Content Engine GUI or the CLI.

**Note**

From the Content Engine GUI, choose **Caching > Content Preload**. Use the displayed Content Preload window to enable and configure this feature on this standalone Content Engine. For more information about how to use the Content Preload window to perform this task, click the **HELP** button in the window.

To use the Content Engine CLI to enable and configure content preloading on a standalone Content Engine, follow these steps:

- 
- Step 1** Enable content preloading on the Content Engine.
- ```
ContentEngine(config)# pre-load enable
```
- Step 2** Create the preload URL list file, as described in the [“Creating a Preload URL List File”](#) section on page 11-3.
- Step 3** Specify the maximum number of concurrent requests for the URL retrieval. You can specify a value from 1 to 30 (for example, 24). The default is 10. If the number of URLs in the preload URL list file is fewer than the number of specified concurrent requests, then the lesser number is active.
- ```
ContentEngine(config)# pre-load concurrent-requests 24
```
- Step 4** Specify the default depth level for the URL retrieval (for example, four levels deep). You can specify a value from 0 to 20. The default is 3. Setting the depth level default to 0 would be useful if you have specified URLs in preload.txt files and you do not want the Content Engine to try to preload other URLs.
- ```
ContentEngine(config)# pre-load depth-level-default 4
```

**Step 5** Specify the path of the file that contains the URL list or a URL.

```
ContentEngine(config)# pre-load url-list-file path
```

*path* is the path of the file that contains the URL list or a URL. For example:

```
pre-load url-list-file /local1/myurllist
pre-load url-list-file ftp://ftpserver/ftpdirectory/urllist.txt
pre-load url-list-file http://server/directory/urllist.txt
```



**Note** The actual preloading of HTTPS links is not supported; only the downloading of the preload URL list file through the HTTPS protocol.

**Step 6** Specify the domains to be fetched during the preload process (for example, cisco.com).

```
ContentEngine(config)# pre-load fetch domain cisco.com
```

**Step 7** Specify that other domains in an HTML page should be traversed. By default, other domains in an HTML page are not traversed during the content preload.

```
ContentEngine(config)# pre-load traverse-other-domains
```

**Step 8** Specify the suffixes to be excluded from the preload operation. This creates a filter for the objects that are to be excluded.

```
ContentEngine(config)# pre-load no-fetch suffix .mil .su .ca
```

**Step 9** Configure a maximum bandwidth for the preloading process (for example, 50,000 kbps).

```
ContentEngine(config)# pre-load max-bandwidth 50000
```



**Note** With the ACNS 5.x software, you can preload WMT streaming media files that may have different bit rates at the URL specified for content preloading. You can also control WMT bandwidth using the **bandwidth wmt outgoing** and **bandwidth incoming** global configuration commands. For more information, see the “[Configuring Incoming and Outgoing WMT Bandwidth and Bit Rates](#)” section on page 9-23.

**Step 10** To trigger a content preload immediately, enter the **pre-load force EXEC** command.

**Step 11** To configure the Content Engine to preload specific content for a future time, use the **pre-load schedule** global configuration command. The Content Engine accesses the specified preload URL list file with a frequency set by the specified preloading schedule (set through the **pre-load schedule** global configuration command or the Content Engine GUI [**Caching > Content Preloading**]).

The default start time for the preloading operation is 00:00 (that is, the start of the day). If the end time is not specified, the preload operation is completed after all the objects have been downloaded. If you want to change this default, do the following:

- a. To specify the start and end times for daily or weekly preloads, use *hh:mm* (where *hh* is the hour, and *mm* is the minutes, for example, 01:00). For hourly preloads, use *mm* to specify the start and end times. The following example shows how to specify a daily interval for scheduling the content preload. In this example, the preload operation starts every day at 1:00 AM and ends every day at 2:00 AM:

```
ContentEngine(config)# pre-load schedule every-day start-time 01:00 end-time 02:00
```

- b. To specify the start time and end times for hourly preloads, the start time should be 0 and the end time should be 59. For daily and weekly preloads, the start time should be from 0 to 23, and the end time should be from 0 to 59. If the endtime option is not specified, the preload operation will continue until completion.

To configure a preload on more than one day of the week, use the **pre-load schedule every-week** global configuration command. The following example shows how to schedule a preload operation every week on Sunday and Wednesday from 1:00 AM to 6:00 AM:

```
ContentEngine#(config)# pre-load schedule every-week Sun Wed
start-time 01:00 end-time 06:00
```

- Step 12** Set the Type of Service (ToS) value as well as the differentiated services code point (DSCP) for all preload traffic by using the **pre-load dscp** global configuration command to set the Type of Service (ToS) value as well as the differentiated services code point (DSCP) for all preload traffic.

Setting the ToS or DSCP is called packet marking, allowing you to partition network data into multiple priority levels or types of service. You can set the ToS or DSCP values in IP packets based on a URL match, a file type, a domain, a destination IP address, a source IP address, or a destination port.

The ACNS 5.x software includes ToS or DSCP support for HTTP, FTP, and WMT preload traffic. Because content preloading is initiated by the Content Engine and not by the requesting client when a connection is made to an origin server, the ToS or DSCP code point on the traffic going toward the server needs to be set before contact is made with the origin server.

The following example shows how to set Type of Service support to normal:

```
ContentEngine(config)# pre-load set-tos normal
```



**Note** Using the **pre-load dscp** global configuration command takes precedence over any use of the Rules Template configuration commands involving DSCP server configurations.

Table 11-1 describes the DSCP values.

**Table 11-1 DSCP Values**

| DSCP Value | Description                     |
|------------|---------------------------------|
| <0-63>     | Valid DSCP value range          |
| af11       | Packets with AF11 dscp (001010) |
| af12       | Packets with AF12 dscp (001110) |
| af13       | Packets with AF13 dscp (001110) |
| af21       | Packets with AF21 dscp (011010) |
| af22       | Packets with AF22 dscp (010110) |
| af23       | Packets with AF23 dscp (010110) |
| af31       | Packets with AF31 dscp (011010) |
| af32       | Packets with AF32 dscp (011110) |
| af33       | Packets with AF33 dscp (011110) |
| af41       | Packets with AF41 dscp (110010) |
| af42       | Packets with AF42 dscp (110110) |
| af43       | Packets with AF43 dscp (110110) |

**Table 11-1 DSCP Values (continued)**

|         |                                               |
|---------|-----------------------------------------------|
| cs1     | Packets with CS1 (precedence 1) dscp (001100) |
| cs2     | Packets with CS2 (precedence 2) dscp (011000) |
| cs3     | Packets with CS3 (precedence 3) dscp (011100) |
| cs4     | Packets with CS4 (precedence 4) dscp (110000) |
| cs5     | Packets with CS5 (precedence 5) dscp (101100) |
| cs6     | Packets with CS6 (precedence 6) dscp (111000) |
| cs7     | Packets with CS7 (precedence 7) dscp (111100) |
| default | Packets with default dscp (000000)            |
| ef      | Packets with EF dscp (101110)                 |

Table 11-2 describes the TOS values.

**Table 11-2 TOS Values**

| <b>TOS Value</b>  | <b>Description</b>                                |
|-------------------|---------------------------------------------------|
| <0-127>           | Valid ToS value range                             |
| critical          | Packets with critical precedence (110)            |
| flash             | Packets with flash precedence (48)                |
| flash-override    | Packets with flash override precedence (64)       |
| immediate         | Packets with immediate precedence (32)            |
| internet          | Packets with internetwork control precedence (96) |
| max-reliability   | Packets with maximum reliable ToS (2)             |
| max-throughput    | Packets with maximum throughput ToS (4)           |
| min-delay         | Packets with minimum delay ToS (8)                |
| min-monetary-cost | Packets with minimum monetary cost ToS (1)        |
| network           | Packets with network control precedence (112)     |
| normal            | Packets with normal ToS (0)                       |
| priority          | Packets with priority precedence (16)             |

**Step 13** View the status of the current preloading operation.

The following example shows the status of the current preloading operation after using the **pre-load set-tos** and **pre-load max-bandwidth** commands:

```
ContentEngine# show pre-load
Preloading is enabled
Number of concurrent sessions: 10
Depth level: 4
URL List File: /local1/url.txt
DSCP: set-tos normal
Max Bandwidth: 50000 Kbps
Previous preloading operation will be continued.
Preload will not traverse other domains.
Fetch Domains:
Fetch Suffix:
```

```

Fetch Directory:
No-fetch Domain:
No-Fetch Suffix:
No-Fetch Directory:
Scheduling on all days
  Start Time: 00:00
  End Time : Till completion

```

- Step 14** View the statistics associated with the current preloading, after the preload has started.

```

ContentEngine# show statistics pre-load
Statistics of last Preloading operation
-----

```

```

Preloading is in progress.
List of preloaded URLs are in /local1/preload_dir/downloaded_urls.

83 objects downloaded, 2842292 bytes transferred.

```

- Step 15** Inform your end users what the URLs of the preloaded files are so that they can use their browsers or media players to access this preloaded content.

---

For an example of how you can verify if the preloaded VOD files are being cached and properly distributed to clients, see the [“Verifying That Preloaded VOD Files Are Cached and Properly Distributed to Windows Media Clients”](#) section on page 9-36.

## Proxy Authentication Support for Content Preloading

Typically preloading works by having a standalone Content Engine preload the content that is specified in the URL list file (the url.txt file). The URL list file also specifies the desired depth level for the URL retrieval (for example, four levels deep). The username and password are available in the URL to perform server authentication if necessary to retrieve a protected object. If there is an intermediate proxy, the preloading of content will not work because the Content Engine is unable to perform proxy authentication.

In releases prior to the ACNS 5.4.1 software release, a typical deployment was that a second Content Engine (CE2) acted as an upstream proxy server for another Content Engine (CE1). CE1 was able to support NTLM authentication if you had specified the **http authentication header 401** global configuration command on CE2. In the case of NTLM authentication, requests would be authenticated (treating them as WWW-Auth), and if the credentials matched, the CE1 would retrieve the contents and the content would be preloaded on CE1.

In the ACNS 5.4.1 software release, proxy authentication support for content preloading was added. Proxy authentication support for content preloading is supported if the Content Engine has been configured to use any of the following authentication schemes for the outgoing proxy server: NTLM, LDAP, RADIUS, and TACACS+. This feature supports one level of upstream proxy server (that is, the proxy information that is obtained from the preload file only applies to the immediate upstream proxy server).

In the ACNS 5.4.1 software and later releases, if CE1 is configured to use NTLM or basic authentication as the proxy authentication scheme, CE1 is able to perform the initial proxy authentication and then retrieve the requested object from the origin server. The retrieved object can be an unauthenticated or authenticated object.

To support proxy authentication in preloading, content preloading on the Content Engine must be able to accept the proxy user and proxy domain name in order to perform proxy authentication for NTLM or Basic authentication. The following URL format for the preloading process is currently supported:

```
http://user1:user1@10.77.157.131/kerberos/kerberos.htm 1 acns:acns:acns
```

where *user1* is the user name, *user1* is the password, 1 is the depth level, *acns* is the domain name, host name, and domain in which the host resides.

The format for accepting the username, password, and domain name for proxy authentication is as follows:

```
proxy_user:username:proxy_pwd:pwd:proxy_domain_name:domain
```

The username, password, and domain name each can be up to 50 characters in length. This information must be specified in the first line of the preload list file. A colon (:) is used as separator to obtain the information from the preload list file. Consequently, do not use a colon in the password for a user.

If the username, password, and domain name for proxy authentication is not available in the preload file, then the Content Engine will use the URL format to perform the preload operation.

If the Content Engine is configured to use NTLM as the proxy authentication scheme, then the preload file would contain the following type of information:

```
proxy_user: user1:proxy_pwd:user1:proxy_domain_name:acns
```

If the Content Engine is configured to use basic authentication as the proxy authentication scheme, then the preload file would contain the following type of information:

```
proxy_user: tuser1:proxy_pwd:tpass1:proxy_domain_name:null
```

For an example of how to configure proxy authentication for content preloading on standalone Content Engines, see the next section, the “[Example of Configuring Proxy Authentication for Content Preloading.](#)”

## Example of Configuring Proxy Authentication for Content Preloading

In the following example, the Content Engine (CE1) is configured to have the Content Engine named CE2 be its outgoing proxy server, and is configured to use NTLM authentication to preload content:

- Step 1** The Content Engine (CE1) is configured to use NTLM authentication as the proxy authentication method:

```
CE1(config)# ip name-server name-server
CE1(config)# ntlm server host ip-address
CE1(config)# ntlm server enable
```

- Step 2** Clear the cache and the statistics on CE1:

```
CE1# clear cache
CE1# clear statistics all
```

- Step 3** Enable the preload feature on CE1:

```
CE1(config)# pre-load enable
```

- Step 4** Configure the path for the url-list file on CE1:

```
CE1(config)# pre-load url-list-file /local1/preload.txt
```

- Step 5** Configure the outgoing proxy server (CE2) for CE1:

```
CE1(config)# http proxy outgoing host ip-address port-number
```

**Step 6** On CE1, enter the following command to preserve the 407 HTTP authentication header:

```
CE1(config)# http proxy outgoing preserve-407
```

**Step 7** On CE2, enable authentication:

```
CE2(config)# http proxy incoming port-number
```

**Step 8** On CE1, in the first line of url-list-file, specify the credentials that are to be matched against the NTLM origin server:

```
CE1# type preload.txt
proxy_user:preload:proxy_pwd:preload1:proxy_domain_name:acns
http://www.yahoo.com/
http://10.77.157.60/
```

In the above preload file, the username is preload, the password is preload1, and the domain name is acns. These credentials will be sent to the NTLM origin server when CE1 receives a 407 message from the outgoing proxy server (CE2).

**Step 9** Force a preload operation on CE1:

```
CE1# preload force
```

**Step 10** After the forced preload operation has been completed on CE1, verify that all of the objects have been preloaded properly on CE1.

- a. On CE1, check the latest\_preloaded\_objects files in the /local/local1/preload\_dir directory.
- b. On CE1, check the latest\_preload\_error file to verify that there are no entries indicating an error because of a proxy authentication failure or a 407 message.
- c. On CE1, enter the following command to verify that the objects have been properly preloaded on CE1:

```
CE1# show statistics preload
Preloading was initiated by force.
Preloading started at Thu Mar 13 06:42:40 2003
Preloading ended at Thu Mar 13 06:42:53 2003
List of preloaded URLs are in /local1/preload_dir/latest_preloaded_objects.
Preload errlog is /local1/preload_dir/latest_preload_error.

Number of invalid entries in URL list file = 0
Total number of preloaded objects = 1
Total number of preloaded bytes = 570
```

- d. On CE2, enter the following command to verify that the user (the user with the name preload) is listed in the authentication cache on CE2:

```
CE2# show http-authcache
```

**Step 11** From a web client that has CE1 as its proxy server, use the browser to issue a request for http://10.77.157.60.

**Step 12** Verify that this client request is served from the cache on CE1:

- a. On CE1, enter the following command:

```
CE1# show statistics http savings
```

- b. Check the command output to verify that the hit counter has been incremented on CE1.

- c. On CE1, enter the following command to verify that the objects have been properly preloaded on CE1:

```
CE1# show statistics preload
Preloading was initiated by force.
Preloading started at Thu Mar 13 06:42:40 2003
Preloading ended at Thu Mar 13 06:42:53 2003
List of preloaded URLs are in /local1/preload_dir/latest_preloaded_objects.
Preload errlog is /local1/preload_dir/latest_preload_error.

Number of invalid entries in URL list file = 0
Total number of preloaded objects = 1
Total number of preloaded bytes = 570
```

- d. On CE2, enter the following command to verify that the user (the user with the name preload) is listed in the authentication cache on CE2:

```
CE2# show http-authcache
```

**Step 13** From a web client that has CE1 as its proxy, use the browser to issue a request for `http://10.77.157.60`.

**Step 14** Verify that this client request is served from the cache on CE1:

- a. On CE1, enter the following command:

```
CE1# show statistics http savings
```

- b. Check the command output to verify that the hit counter has been incremented on CE1.

## Stopping or Resuming Content Preloading on Standalone Content Engines

To stop a preload process that is currently in progress on a standalone Content Engine, use the **no pre-load enable** global configuration command.

If the content preloading is not completed before the scheduled end time, you can resume the preloading process to capture content using the **pre-load resume** global configuration command. Using this command allows you to resume downloading from the breakpoint of the previous preload, instead of starting again from the very beginning of the preload URL list file.



### Note

If the **pre-load resume** global configuration command is not set up on the Content Engine and content preloading is aborted before the scheduled end time, the next scheduled content preloading starts from the beginning of the preload URL list file.

## Configuring URL Filtering on Standalone Content Engines

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

[Table 11-3](#) lists the various URL filtering schemes that you can configure a standalone Content Engine to use in order to control client access to websites.

**Table 11-3** URL Filtering Methods with Standalone Content Engines

| URL Filtering Scheme  | More Information                                                                                                                                                                                                                                |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Local list files      | Deny access to URLs specified in a list. Permit access only to URLs specified in a list. See the <a href="#">“Configuring Local List URL Filtering on Standalone Content Engines”</a> section on page 11-13.                                    |
| N2H2 external servers | Direct client requests for content to an external N2H2 server for URL filtering. See the <a href="#">“Configuring Standalone Content Engines for N2H2 URL Filtering”</a> section on page 11-18.                                                 |
| Websense server       | Direct client requests for content to either the local Websense plug-in or to an external Websense server for URL filtering. See the <a href="#">“Configuring Standalone Content Engines for Websense URL Filtering”</a> section on page 11-21. |
| SmartFilter plug-in   | Direct client requests for content to the SmartFilter plug-in for URL filtering. See the <a href="#">“Configuring URL Filtering with SmartFilter Software”</a> section on page 11-40.                                                           |

See [Table B-6](#) for a list of the URL filtering schemes (for example, SmartFilter or Websense) that are supported for different protocols.

Although only one form of URL filtering scheme can be active at a time per protocol, many URL filtering schemes can be supported at one time. For example, if an N2H2 filter is applied to HTTP requests, then no other URL filtering scheme (for instance, Websense or SmartFilter) can be applied to this protocol. However, you could apply the local list URL filtering scheme (good lists and bad lists) to the streaming media protocols (WMT client requests and client requests over RTSP). The scheme enabled for a particular protocol is independent from that of other protocols.

**Note**

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

To ensure that URL filtering applies to every URL that passes through the Content Engine, disable all bypass features. By default, load bypass is enabled.

- To use the Content Engine GUI to disable load bypass manually, choose **Caching > Bypass** and then click the **Load Bypass Off** radio button in the Bypass window.
- To disable load bypass manually through the Content Engine CLI, use the **bypass load** global configuration command.

```
ContentEngine(config)# no bypass load enable
```

- To disable error handling manually through the Content Engine CLI, use the **error-handling send-cache-error** or **error-handling reset-connection** global configuration command. (By default, error handling is enabled on the Content Engine.)

```
ContentEngine(config)# no error-handling send-cache-error
ContentEngine(config)# no error-handling reset-connection
```

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

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

```
test          Password = "test"
              Service-Type = Framed-User,
              Filter-Id = "No-Web-Blocking"
```

The Filter-Id attribute is defined as either No-Web-Blocking or Yes-Web-Blocking. Yes-Web-Blocking means that the request is subject to URL filtering, and No-Web-Blocking means that the request is not subject to URL filtering. If blocking is not specified, Yes-Web-Blocking is the default RADIUS filter.

**Note**

For more information about the use of a RADIUS server for authentication purposes and URL filtering, see the [“Understanding RADIUS Authentication and Authorization”](#) section on page 17-6.

## Configuring Local List URL Filtering on Standalone Content Engines

You can configure standalone Content Engines to deny client requests for URLs that are listed in a badurl.lst file, or configure them to fulfill only requests for URLs in a goodurl.lst file. The use of local list files (URL lists) applies to HTTP (HTTP, HTTPS-over-HTTP, and FTP-over-HTTP) as well as RTSP streaming media protocol. This type of URL filtering is referred to as *local list URL filtering*.

**Tip**

Only one good sites file or one bad sites file can be active at one time per protocol.

The local list file for each protocol should not contain URLs that belong to other protocols. For instance, the HTTP local list file should contain only the following types of URLs: HTTP, HTTPS, or FTP URLs. In the ACNS 5.3.1 software and later releases, the WMT local list file can contain RTSP URLs.

**Caution**

If the size of the local list file becomes too large, it can adversely effect proxy performance, because the local list file is loaded into memory when local list filtering is enabled. If the file size is larger than 5 MB, a warning message appears, but the ACNS software does not enforce size limits for the local list file. It is your responsibility to track the local list file size and ensure that it does not become so large that it degrades performance.

You can configure a standalone Content Engine to use local list URL filtering to filter the following types of client requests for content:

- Requests over HTTP (HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests)
- RealMedia requests (IETF standard RTSP protocol with RealNetworks proprietary extensions)
- WMT requests (MMS-over-HTTP and RTSP-over-RTP for Windows Media 9 clients and Windows Media 9 servers)

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

Support for RTSP-over-RTP (as referred to as *WMT RTSP requests*) for Windows Media 9 players is available in the ACNS 5.3.1 software and later releases. For WMT RTSP requests, there are three possible protocol prefixes: rtsp, rtspu, and rtspt.

If the user enters rtsp: as the protocol prefix for the URL, the Windows Media 9 player can choose to use RTSPT or RTSPU. If the rtsp bad file has a URL of `rtsp://hostname/pathname` and the user's URL request is `rtspt://hostname/pathname`, then the RTSP request from a Windows Media 9 player might get through the URL filtering. Special URL filtering for RTSP requests from Windows Media 9 players is available in the ACNS 5.3.1 software and later releases.

For WMT URL filtering, filtering for RTSP URLs (`rtsp://`) is only supported; there is no separate filtering support for RTSPT and RTSPU URLs. However, if you configure an RTSP URL in the badurl.lst file, then it will block both the RTSPT and RTSPU URLs.

To use the Content Engine CLI to configure local list URL filtering on a standalone Content Engine, use the **url-filter** global configuration commands. In the ACNS 5.3.1 software release, the **url-filter** command was modified to support local list URL filtering for RTSP requests from Windows Media 9 players. In the ACNS 5.2.x software and earlier releases, the **url-filter** command options were as follows:

```
ContentEngine(config)# url-filter ?
  http  For requests over HTTP
  rtsp  For requests over RTSP
  wmt   For WMT requests
```

In the ACNS 5.3.1 software and later releases, the **url-filter** command options are as follows:

```
ContentEngine(config)# url-filter ?
  http  For requests over HTTP and MMS over HTTP
  rtsp  For requests over RTSP - applies to real proxy, real server and cisco
        streaming engine
  wmt   For WMT requests - applies to MMS and RTSP
```

Local list URL filtering is the only supported filtering method for WMT requests (MMS-over-HTTP and RTSP requests from WMT clients) and RTSP requests (requests from RealMedia players). The third-party URL filtering methods (N2H2, SmartFilter, and Websense software) are not supported for WMT and RTSP requests. For HTTP requests, the local list URL filtering method as well as N2H2, SmartFilter, and Websense URL filtering is supported. See [Table B-4](#) for a list of the kinds of URL filtering that are supported for the different protocols.

[Table 11-4](#) describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for HTTP requests (HTTP, FTP-over-HTTP, MMS-over-HTTP, and HTTPS-over-HTTP requests).

**Table 11-4** Configuring Standalone Content Engines to Use Local List URL Filtering for Requests over HTTP

| CLI Command                                                  | Description                                                                                 |
|--------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <b>url-filter http bad-sites-deny enable</b>                 | Configures the Content Engine to deny client requests to URLs in the HTTP bad site list.    |
| <b>url-filter http bad-sites-deny file <i>filename</i></b>   | Specifies the filename of the HTTP bad site list.                                           |
| <b>url-filter http good-sites-allow enable</b>               | Configures the Content Engine to permit client requests to URLs in the HTTP good site list. |
| <b>url-filter http good-sites-allow file <i>filename</i></b> | Specifies the filename of the HTTP good site list.                                          |

Table 11-5 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for requests over RTSP. This type of URL filtering is used with RealProxy, which is a backend RTSP server that is running on the standalone Content Engine. With registered Content Engines, RealProxy, RealSubscriber, and the Cisco Streaming Engine that are running on the registered Content Engine use this type of URL filtering.

**Table 11-5** Configuring Standalone Content Engines to Use Local List URL Filtering for Requests over RTSP

| CLI Command                                                  | Description                                                                                 |
|--------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <b>url-filter rtsp bad-sites-deny enable</b>                 | Configures the Content Engine to deny client requests to URLs in the RTSP bad site list.    |
| <b>url-filter rtsp bad-sites-deny file</b> <i>filename</i>   | Specifies the filename of the RTSP bad site list.                                           |
| <b>url-filter rtsp good-sites-allow enable</b>               | Configures the Content Engine to permit client requests to URLs in the RTSP good site list. |
| <b>url-filter rtsp good-sites-allow file</b> <i>filename</i> | Specifies the filename of the RTSP good site list.                                          |

Table 11-6 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for WMT requests (WMT requests over RTSP). If you configure an RTSP URL in a WMT bad site list, then it blocks both the RTSP and RTSPU URLs as well as the RTSP URL that is specified in the bad site list.

**Table 11-6** Configuring Standalone Content Engines to Use Local List URL Filtering for WMT Requests

| CLI Command                                                 | Description                                                                                |
|-------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| <b>url-filter wmt bad-sites-deny enable</b>                 | Configures the Content Engine to deny client requests to URLs in the WMT bad site list.    |
| <b>url-filter wmt bad-sites-deny file</b> <i>filename</i>   | Specifies the filename of the WMT bad site list.                                           |
| <b>url-filter wmt good-sites-allow enable</b>               | Configures the Content Engine to permit client requests to URLs in the WMT good site list. |
| <b>url-filter wmt good-sites-allow file</b> <i>filename</i> | Specifies the filename of the WMT good site list.                                          |

In the ACNS 5.3.1 software and later releases, the **url-filter wmt** global configuration commands apply to RTSP. URL filtering for RTSP requests is used when the client is a Windows Media 9 player and the server is a Windows Media 9 server. If an earlier version of the Windows Media player is used (for example, Windows Media 7 players), the MMS-over-HTTP protocol is used instead of the RTSP protocol to service the content request from the Windows Media player.

## Example of Configuring URL Filtering with Local URL Lists

To configure a standalone Content Engine to use a local list file to deny client requests for specific HTTP URLs, follow these steps:

---

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

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

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




---

**Tip** We recommend creating a separate directory under local1 to hold the bad lists, for example, `/local1/filtered_urls`.

---

**Step 3** Point the Content Engine to the bad URL list.

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

**Step 4** Configure the Content Engine to actively deny the URLs.

```
ContentEngine(config)# url-filter http bad-sites-deny enable
```

**Step 5** Reload the new bad site list on the standalone Content Engine.

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

---

To configure a standalone Content Engine to use a local list file permit specific HTTP URLs to the exclusion of all other URLs, follow these steps:

---

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

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

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




---

**Tip** We recommend creating a separate directory under local1 to hold the good lists, for example, `/local1/filtered_urls`.

---

**Step 3** Point the Content Engine to the goodurl.lst file.

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

**Step 4** Configure the Content Engine to actively permit only the good URLs.

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

**Step 5** Reload the new good site list on the standalone Content Engine.

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

---

## Reloading Local List Files on Standalone Content Engines

When you update the badurl.lst or goodurl.lst file, use the **url-filter local-list-reload EXEC** command to reload the good site or bad site lists on the standalone Content Engine if the URL list feature is enabled.

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

The syntax is as follows:

- **http** reloads the new local lists for HTTP requests (HTTP, FTP-over-HTTP, MMS-over-HTTP, and HTTPS-over-HTTP requests).
- **rtsp** reloads the local lists for requests over RTSP (requests from RealMedia clients).
- **wmt** reloads the local lists for WMT requests (RTSP-over-RTP [the standard IETF RTSP protocol with Microsoft proprietary extensions] requests from Windows Media 9 players).

The following examples shows how to reload new good site or bad site lists on a standalone Content Engine:

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

## Creating Custom Blocking Messages

In the case of local list URL filtering, you can configure standalone Content Engines to return a customized blocking message to the client that requested content that is served through the Content Engine. The custom message must be an administrator-created HTML page named block.html. Make sure to copy all embedded graphics associated with the custom message HTML page to the same directory that contains the block.html file. The following is an example of the contents of the block.html file:

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

If the block.html file is updated, it will automatically display its new message without requiring you to reenter the **url-filter http custom-message** command.

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

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

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

To enable the customized blocking message, use the **`url-filter http custom-message`** global configuration command and specify the directory name. To disable the custom message, use the **`no url-filter http custom-message`** command.

You can enable and disable the **`url-filter http custom-message`** command without affecting the **`good-sites-allow`** and **`bad-sites-deny`** configuration.


**Note**

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

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

## Configuring Standalone Content Engines for N2H2 URL Filtering

N2H2 is a globally deployed URL-filtering solution that can filter HTTP, FTP, or HTTPS requests based on destination hostname, destination IP address, and username and password. N2H2 relies on a sophisticated URL database exceeding 15 million sites and is organized into over 40 categories using both Internet technology and human review. Refer to the *Filtering N2H2 Installation and Configuration Guide* for further information on N2H2 filtering products.


**Note**

See [Table B-4](#) for a list of the kinds of URL filtering that are supported for the different protocols with an N2H2 server.

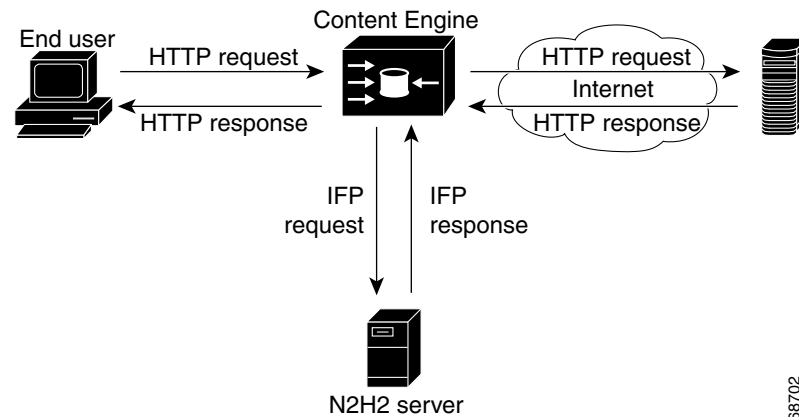
N2H2 supports three filtering methods. [Table 11-7](#) lists the N2H2 features supported by the Content Engine. One N2H2 server can support multiple Content Engines simultaneously.

**Table 11-7** Supported N2H2 Features

N2H2 Feature	Description
Global filtering	Applies filtering to all HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over HTTP requests.)
User-based filtering	Applies filtering to specific users or groups.
Client IP-based filtering	Applies filtering to specific client IP addresses.
Transparent authentication	Performs transparent authentication by passing back the initial response header to the client using the HTML page in IFP responses.

Standalone Content Engines can use an N2H2 enterprise server as a filtering engine and enforce the filtering policy configured on the N2H2 server. (See [Figure 11-1](#).) The standalone Content Engine and the N2H2 server use Internet Filtering Protocol (IFP) Version 2 to communicate with each other. When the Content Engine receives a URL request, it sends an IFP request to the N2H2 server with the requested URL. The N2H2 server does some necessary lookups for the URL and sends back an IFP response. Based on the N2H2 server's IFP response, the Content Engine either blocks the HTTP request by redirecting the browser to a page where a blocking message is displayed or proceeds with normal HTTP processing by sending the URL request to an origin server.

**Figure 11-1 N2H2 Filtering**



**Note**

URL filtering using an N2H2 server is applied to HTTP traffic (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) before the Rules Template is applied, regardless of whether the requested object is in the cache or not. See [Table B-4](#) for the kinds of URL filtering that are supported for the different protocols with N2H2.

To configure a standalone Content Engine to use an external N2H2 server for URL filtering, follow these steps:

- Step 1** Display the URL filtering schemes that are currently enabled on this Content Engine for requests over HTTP.
- ```
ContentEngine# show url-filter http
```
- Step 2** Make sure that no other URL filtering scheme (for example, Websense or SmartFilter software) is currently enabled for requests over HTTP. Only one URL filtering scheme per protocol can be active at a time.

**Step 3** Configure the Content Engine to use an external N2H2 server for URL filtering by using the **url-filter http N2H2 server** global configuration command.

- a. Specify the necessary information about the external N2H2 server (for example, its IP address).

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

- *hostname* is the hostname of the external N2H2 server.
- *IP address* is the IP address of the external N2H2 server.
- *portnum* is the port number (1–65535) to which the Content Engine sends the IFP requests to the specified N2H2 server. The default port number is 4005.
- *seconds* is the number of seconds (1–120) that the Content Engine is to wait for an IFP response from the N2H2 server before timing out the connection. The default timeout is 5 seconds.

In the following example, the Content Engine is configured to use an N2H2 server that has an IP address of 172.16.22.10. The Content Engine will send IFP requests to this N2H2 server on port 4008 and will wait for up to 100 seconds for an IFP response from this server before timing out the connection:

```
ContentEngine(config)# url-filter http N2H2 server 172.16.22.10 port 4008 timeout 100
```

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



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

**Step 4** Enable the N2H2 URL filtering scheme on this Content Engine.

```
ContentEngine(config)# url-filter http N2H2 enable
```

**Step 5** Allow HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) to pass through when the N2H2 server is enabled but the Content Engine has problems communicating with the N2H2 server by using the **url-filter http N2H2 allowmode enable** global command.

- When **allowmode** is enabled, the Content Engine allows all HTTP traffic to continue through it (it proceeds with normal traffic processing) even if it fails to receive responses from the N2H2 server.
- When **allowmode** is disabled, the Content Engine blocks all HTTP traffic that is served through it if it fails to receive responses from the N2H2 server.

By default, **allowmode** is enabled. You can configure the **allowmode** option with or without N2H2 being enabled; it is independent of the N2H2 server configuration. The Content Engine adopts the new configuration for **allowmode** if N2H2 URL filtering is already being used.

**Step 6** Display the request-reply statistics for the communication between the Content Engine and the N2H2 server.

```
ContentEngine# show statistics url-filter http N2H2
```

These statistics show the number of requests sent, replies received, pages blocked, pages allowed, and failure cases. More detailed URL filtering statistics are available on the N2H2 server. The statistics shown can be cleared using the **clear statistics url-filter http N2H2** and **clear statistics all EXEC** commands. The **clear statistics url-filter http N2H2 EXEC** command resets the statistics counters for the N2H2 server. All the statistics counters are reset to 0.

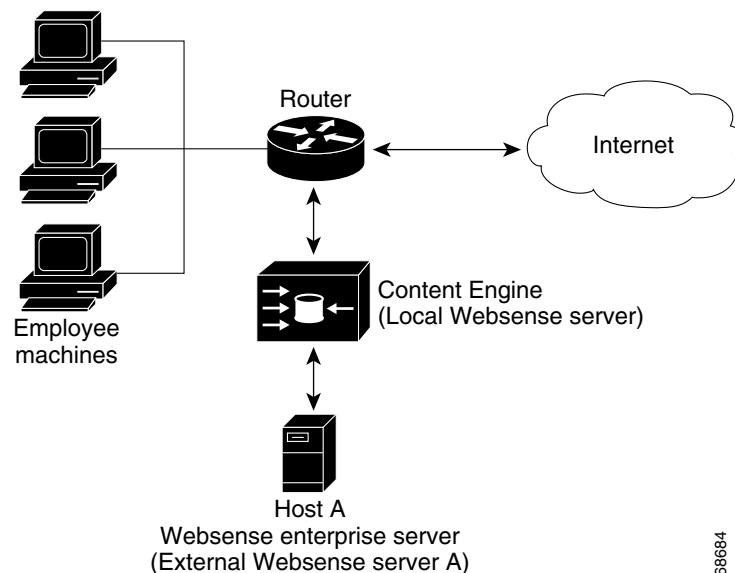
**Note**

Refer to the Filtering N2H2 Installation and Configuration Guide for more information about N2H2 filtering configuration and policies.

## Configuring Standalone Content Engines for Websense URL Filtering

Standalone Content Engines can use a remote Websense enterprise server as a filtering engine and enforce the filtering policy configured on the Websense server. As [Figure 11-2](#) shows, the remote Websense server runs on a separate system (Host A) from the local Websense server and communicates with the standalone Content Engine over the network.

**Figure 11-2** URL Filtering with Websense Servers



You can also configure a standalone Content Engine to use the integrated Websense server. The integrated Websense server is an internal server that runs on the Content Engine, and is referred to as the *local Websense server*.

**Note**

In the ACNS 5.1 software and earlier releases, only one Websense server is supported. In the ACNS 5.2.1 software and later releases, up to two Websense servers are supported. For more information on this topic, see the next section, “[About Websense Server Failover](#).”

The ACNS 5.4.1 software and later releases support the Websense 5.5.2 software. The ACNS 5.3.x software supports the Websense 5.2.0 software. The ACNS 5.5.5 and later software supports the Websense 6.2 software.

## About Websense Server Failover

In the ACNS 5.2.1 software and later releases, the Websense server failover feature is supported. This feature allows you to configure a Content Engine to use up to two Websense servers for failover purposes (one primary and one secondary server) for URL filtering. [Table 11-8](#) lists the supported Websense server failover configurations.

**Table 11-8 Supported Websense Server Failover Configurations**

| Supported Configurations | Local (Internal) Websense Server                                      | Remote Websense Server   |
|--------------------------|---|--|
| Option A                 | The local Websense server is disabled on the Content Engine.          | The primary Websense server is running on an external host (for example, Host A).<br>The secondary Websense server is running on a second external host (for example, Host B). |
| Option B                 | The local Websense server is acting as the primary Websense server.   | The secondary Websense server is running on an external host.  |
| Option C                 | The local Websense server is acting as the secondary Websense server. | The primary Websense server is running on an external host.  |

The order in which you configure the Websense servers determines which server is designated the primary Websense server. The first configured Websense server is designated the primary server. Configuration of a secondary Websense server is optional. For an example of how to configure Websense server failover for a standalone Content Engine, see the [“Example of Configuring Websense Server Failover and URL Filtering”](#) section on page 11-30.

## About Websense Services

The ACNS 5.4.x software and later releases supports the Websense 5.5.2 software. The following services are supported in the Websense 5.5.2 software:

- Policy Server
- Employee Internet management (EIM) Server
- Local Network Agent
- Local RADIUS Agent
- Local eDirectory Agent
- Local Logon Agent (support added in the ACNS 5.4.1 software release)
- Local User Service



### Note

When you install or activate additional Websense components for the integrated (local) Websense server, the ACNS software requires a minimum of 1 GB of RAM.

**Note**

The term *local Websense server* is still used to refer collectively to these Websense processes that are running internally on the Content Engine. These Websense processes are also called *services*.

In the Websense GUI Manager Version 5.5 and 6.1 that is supported by the Websense 5.5.2 software, you can configure the RADIUS and eDirectory agent through the GUI as well as through CLI commands. Consequently, in the ACNS 5.4.1 software release, all of the global configuration CLI commands that were related to configuring the RADIUS and eDirectory agent have been removed from the ACNS CLI command set. However, the CLI commands that are used to activate the RADIUS and eDirectory agents have been retained (the **websense-server service radius-agent activate** and the **websense-server service edirectory-agent activate** global configuration commands).

- The following global configuration CLI commands for configuring the RADIUS agent were removed in the ACNS 5.4.1 software release:
  - **websense-server service radius-agent incoming** [*auth-number*] [**acct-port** *port number*]
  - **websense-server service radius-agent outgoing** [*host remote-RADIUS-server IP-address*] [**auth-port** *port number*] [**acct-port** *port number*]
- The following global configuration CLI commands for configuring the eDirectory agent were removed in the ACNS 5.4.1 software release:
  - **websense-server service edir-agent edir-server** [**administrative-dn** *administrative-distinguished-name*] [**host** *remote-eDirectory-server IP-address*] [**root-context** *root-context*]

If the Content Engine is running the ACNS 5.4.1 software or a later release, and if you attempt to configure the RADIUS agent or the eDirectory agent through the CLI by entering any of the preceding global configuration commands, the command is nullified. You will not receive an error message if you enter one of these nullified commands.

The global configuration command for configuring the eDirectory administrative password was retained in the ACNS 5.4.1 software release:

```
ContentEngine(config)# websense-server service edirectory-agent edir-server
administrative-passwd password
```

With the Websense 5.5.2 software, you can use a local or remote Websense Policy Server to activate the local EIM Server, the local RADIUS Agent, the local eDirectory Agent, the local Network Agent, the local Logon Agent, and the local User Service individually on a Content Engine. (See [Table 11-9](#).)

**Table 11-9 Services of the Local Websense 5.5.2 Server Supported in the ACNS 5.4.1 Software or Later**

| Name                | Description   |
|---------------------|---|
| Policy Server       | <p>Hosts all of the policy information that you have configured through the external Websense Manager GUI. Communicates the policy information to the other services of the local Websense server (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service).</p> <p>The local (internal) Policy Server or the specified remote Policy Server must be running, before you can activate the local EIM Server, the local Network Agent, the local Logon Agent, or the local User Service on the Content Engine.</p>   |
| Local EIM Server    | Provides the URL filtering functionality when used with proxy servers, firewalls, and caching appliances.   |
| Local Network Agent | <p>Enables URL filtering of requests that use protocols other than HTTP, HTTPS-over-HTTP, and FTP-over HTTP. If the local Network Agent is activated on the Content Engine, the Network Agent can filter incoming requests from the following protocols and applications:</p> <ul style="list-style-type: none"> <li>• Database applications such as SQL Net</li> <li>• File transfer applications such as FTP and Gopher</li> <li>• Instant messaging and chat applications such as Yahoo Messenger, and MSN Messenger</li> <li>• Mail and collaborative tools such as POP3, SMTP, and NetMeeting</li> <li>• Network operating system applications such as Daytime, finger, NTP, SSH, and Telnet</li> <li>• Remote access applications such as VNC and pcANYWHERE</li> <li>• Streaming media applications such as RTSP, Windows Media, and Liquid Audio</li> <li>• Other (for example, Network News Transfer Protocol [NNTP])</li> </ul> |
| Local RADIUS Agent  | <p>Enables URL filtering based on user-based or group-based policies for users who are authenticated through an external RADIUS server. This agent transparently identifies the users who access the network and are authenticated through the RADIUS authentication scheme. When the Content Engine is supplied with this information, the Content Engine can apply policies to users and groups of the users who access the network remotely.</p> <p>This agent acts as a proxy that forwards the RADIUS messages between the RADIUS client and the external RADIUS server. For the local RADIUS Agent to work properly, you must have configured the RADIUS settings (for example, the IP address of the external RADIUS server) on the Content Engine, as described in the <a href="#">“Specifying RADIUS Authentication Settings for Standalone Content Engines”</a> section on page 17-10.</p>                                      |

**Table 11-9 Services of the Local Websense 5.5.2 Server Supported in the ACNS 5.4.1 Software or Later (continued)**

| Name                   | Description  |
|------------------------|--|
| Local eDirectory Agent | <p>Enables URL filtering based on user-based or group-based policies for users who are authenticated through LDAP. This agent works in conjunction with the Novell eDirectory to transparently identify users who access the network and are authenticated through the LDAP authentication scheme. When the Content Engine is supplied with this information, the Websense filtering service can filter requests based on the policies applied to the users or groups.</p> <p>This agent uses LDAP to gather the user login session information from the Novell eDirectory, which authenticates users logging in to the network. This agent associates each authenticated user with the IP address. With the help of the Websense local User Service, the local eDirectory Agent supplies this information to the Websense filtering service. For this local eDirectory Agent to work properly, you must have configured settings such as the administrative distinguished name, using the Websense Manager GUI (Version 5.5, or 6.1 or later).</p>  |
| Local User Service     | <p>Enables URL filtering based on user-based or group-based policies. If you are using a user service and you want to configure user-based or group-based URL filtering using a Windows NT directory, then you must use the external user service on a Windows machine.</p>  |
| Local Logon Agent      | <p>Enables URL filtering based on user-based or group-based policies for users as they log on to the network through a Windows client machine. This agent's associated logon application captures logon sessions as users log on to Windows domains in a network. The Logon Agent communicates with the Websense User Service to provide up-to-date user logon session information to Websense for filtering purposes.</p> <p>The Logon Agent identifies users in a real-time manner, as they log on to a domain, whereas a Domain Controller (DC) Agent identifies users by periodically querying domain controllers and workstations. Because the Logon Agent identifies users in a real-time manner, the Websense Filtering Service can accurately filter Internet access based on the policies assigned to particular users, groups, workstations, or networks.</p> <p>The Logon Agent user identification process is the identification process that the Logon Agent relies on for the self-identification of the Windows client machine. Typically, a logon script in a shared network location evokes a process on the client machines called LogonApp.exe. LogonApp.exe has two operation modes: persistent mode and nonpersistent mode.</p> <ul style="list-style-type: none"> <li>• Persistent mode: LogonApp.exe runs as a background task on the client machine, and periodically sends the username and password pairs to the Logon Agent. The interval is determined by the Query Interval (persistent mode) setting on the Websense Enterprise Manager. If a logout script has been configured to register when a user logs out, then the LogonApp.exe sends the logout information to the Logon Agent at that time.</li> <li>• Nonpersistent mode: LogonApp.exe contacts the Logon Agent once when a user logs on. The user logon session is stored in the Logon Agent's local memory. The Logon Agent's user map is subject to the Entry Lifetime (non-persistent mode) setting on the Websense Enterprise Manager.</li> </ul> <p>The Logon Agent obtains logon session information from its associated logon application, LogonApp.exe, and stores the username and IP address pairs in a user map in local memory and in the AuthServer.journal file. IP addresses, rather than usernames, are the key element in tracking logon sessions because it is possible for the same user to log on to a network domain from multiple workstations. Each time the size of the AuthServer.journal file reaches 1 MB, the Logon Agent backs up the contents to the AuthServer.bak file on to its hard disk.</p> <p>The Logon Agent, like the eDirectory agent, transparently identifies users. Support for the Logon Agent was added in the ACNS 5.4.1 software release.</p> |

The Websense 5.5.2 software, which is supported in the ACNS 5.4.1 software and later releases, includes an additional agent called the logon agent.

Table 11-10 lists the CLI commands that are related to configuring the Websense 5.5.2 software on standalone Content Engines.

**Table 11-10** Websense Server-Related CLI Commands

| CLI Command Syntax   | Description   |
|--|---|
| <b>websense-server service policy local activate</b>   | Activates the local Policy Server on the Content Engine.  |
| <b>websense-server service policy remote</b><br>[ <i>host remote-policy-server IP-address</i> ]<br>[ <i>port remote-policy-server port-number</i> ]] | Specifies the remote Policy Server to be used to activate the local EIM Server, the local Network Agent, and the local User Service on the Content Engine. The default port number is 55806.  |
| <b>websense-server service eim activate</b>  | Activates the local EIM Server on the Content Engine. Use the <b>no</b> form of this command to deactivate it.  |
| <b>websense-server service network-agent activate</b>  | Activates the local Network Agent on the Content Engine. Use the <b>no</b> form of this command to deactivate it.   |
| <b>websense-server service user activate</b>   | Activates the local User Service on the Content Engine. Use the <b>no</b> form of this command to deactivate it.  |
| <b>websense-server service radius-agent activate</b>   | Activates the local RADIUS Agent on the Content Engine. Use the <b>no</b> form of this command to deactivate it.  |
| <b>websense-server service edir-agent activate</b>   | Activates the local eDirectory Agent on the Content Engine. Use the <b>no</b> form of this command to deactivate it.  |
| <b>websense-server service edir-agent edir-server administrative-passwd password</b>   | Specifies the administrative password that the Content Engine uses to contact the external eDirectory Server to request a database search.<br><br>The following example shows how to use the <b>administrative-passwd</b> command option to specify default244 as the administrative password:<br><br><pre>ContentEngine(config)# websense-server service edir-agent edir-server administrative-passwd default244</pre> |
| <b>websense-server service logon-agent activate</b>  | Activates the local Logon Agent on the Content Engine. Use the <b>no</b> form of this command to deactivate it. This command was added in the ACNS 5.4.1 software release.  |



**Note**

In the ACNS 5.2 software, the **websense-server ip-address** and **websense-server user-server external** global configuration commands are deprecated.

In the ACNS 5.2.1 software and later releases, you can configure a Content Engine to use up to two Websense servers for URL filtering.

To configure a Content Engine for Websense URL filtering, determine the type of Websense server configuration that you want to use for URL filtering:

- To use two Websense servers (the local Websense server and an external Websense server, or two external Websense servers), see the “[Example of Configuring Websense Server Failover and URL Filtering](#)” section on page 11-30.
- To use only the local Websense server, see the “[Configuring URL Filtering with a Local Websense Server](#)” section on page 11-34.
- To use only an external Websense server, see the “[Configuring Websense URL Filtering with External Websense Servers](#)” section on page 11-37.

Only one URL filtering scheme per protocol can be active at a time. In order to enable Websense URL filtering for requests over HTTP, you should make sure that no other URL filtering scheme is configured per protocol. To display the URL filtering schemes that are currently enabled on this Content Engine for HTTP requests (HTTP, FTP-over-HTTP, and HTTPS-over-HTTP), use the **show url-filter http** EXEC command. See [Table B-4](#) for the kinds of URL filtering that are supported for different protocols with a Websense server.

**Note**

In the ACNS 5.4.x software and later releases, Websense server Version 5.5.2 is supported on all Cisco Content Engine platforms. For more detailed information about configuring the Websense software, go to the following website: <http://www.websense.com/content/home.aspx>

Websense software provides an image of the Websense server that resides in the /local1/WebsenseEnterprise/EIM directory on the Content Engine. All the executables as well as the configuration and logging files are stored in this directory.

In the ACNS 5.4.1 software and later releases, Websense installation output is logged to a file named ws\_history.log. This particular log file resides in the /local1/logs/urlfilter/websense directory on the Content Engine.

In the ANCS 5.4.1 software and later releases, you can enter the **clear websense** EXEC command to remove the existing Websense configuration. This command can be useful if you want to remove a corrupted Websense configuration from a standalone Content Engine.

When the Websense server is enabled and the Websense URL database is downloaded to the Content Engine for the first time, CPU usage will be high. Therefore, it is recommended that you enable the Websense server during off-peak times or at times of low network traffic; otherwise, other processes running on the Content Engine may be affected. If one of the Websense processes exits, the local Websense server is automatically restarted on the Content Engine.

To download the Websense components, such as Explorer, Manager, and Reporter, or to obtain an evaluation key for use with the local Websense server that runs on the standalone Content Engine, access the following URL and follow the sequence of steps:

<http://www.websense.com/content/support.aspx>

## Configuring Ports for the Websense Server

The Websense process requires that four ports be open for connections either from processes internal to the Content Engine or from external processes such as the Websense Manager. (See [Table 11-11](#).)

**Table 11-11** Configuration of Ports for the Websense Server

| Port                               | Description   | Default |
|------------------------------------|---|---------|
| Websense server port               | This is the TCP port that receives requests for content filtering according to the Websense protocol.   | 15868   |
| Block message server port          | If the Websense process blocks a URL, it sends a redirect URL to the user. The redirect URL is configured to print out the blocked page and policy for the user. The Websense process listens on this port to receive the pages blocked, serviced by a thread in the Websense server. This thread sends the blocked page in response to the redirected request.     | 15871   |
| Diagnostics server port            | The Websense server has an exhaustive set of diagnostics that users can run remotely to diagnose problems in the Websense process. This is the port to which these diagnostics utilities connect.   | 15869   |
| Websense configuration server port | This is the port for the Websense Policy Server that the Websense GUI Manager connects to. There is no default entry in the websense.ini file for this port, and we recommend that you do not modify this default.  | 55806   |
| Logon Agent port                   | This is the port that the Logon Agent, which is running on the Content Engine, listens on for communication from LogonApp.exe, which is running on the Windows client machines. The Logon Agent receives the username, hashed password, and the client machine's IP address on this port. Support for the Logon Agent was added in the ACNS 5.4.1 software release. | 15880   |

You can configure the first three ports that are listed in [Table 11-11](#) by modifying the eimserver.ini file that resides in the /local1/WebsenseEnterprise/EIM directory on the standalone Content Engine. The Websense server must be restarted so it can pick up the newly configured ports.

You can modify the ports by exporting a copy of the eimserver.ini file using FTP from the /local1/WebsenseEnterprise/EIM directory on the Content Engine, modifying the file, deleting the eimserver.ini file on the Content Engine, and then sending back the modified file to the Content Engine using FTP.



### Note

The Websense server needs to be disabled and then reenabled to pick up newly configured ports. To disable the local Websense server, use the **no websense-server enable** global configuration command. Also make sure that you use the **url-filter http websense server** global configuration command to point the Websense client to the correct Websense server port. For more information about the **url-filter http websense server** command, see the *Cisco ACNS Software Command Reference, Release 5.5* publication.

## Websense Issues When Upgrading to ACNS 5.4.x Software

If you upgrade from the ACNS 5.3.x software to the ACNS 5.4.x software, support for the RADIUS and eDirectory configurations will no longer be provided by the Websense binaries. Consequently, the global configuration commands that were used to configure the RADIUS and eDirectory agents are deprecated in the ACNS 5.4.1 software release.

If you have used the CLI configuration commands to configure the RADIUS and eDirectory agents on a Content Engine that is running the ACNS 5.3.x software, and you upgrade from the ACNS 5.3.x software to the ACNS 5.4.x software, the configuration for the RADIUS and eDirectory agents that were stored in the `wradius.ini` and `wsedire.ini` files will be retained. However, any configuration changes that are made to the RADIUS and eDirectory agents through the Websense GUI Manager will not be reflected in these two `.ini` files.



### Note

The CLI commands that are used to activate the RADIUS and eDirectory agents (the **`websense-server service radius-agent activate`** and **`websense-server service edirectory-agent activate`** global configuration commands) and to specify the eDirectory administrative password (the **`websense-server service edirectory-agent edir-server administrative-passwd password`** global configuration command) have been retained in the ACNS 5.4.x software.

If you are upgrading from the ACNS 5.2.x software to the ACNS 5.4.x software, you do not have to make any changes to the previous Websense configuration because the RADIUS and eDirectory Agents were not supported in the ACNS 5.2.x software release.

## Websense Issues When Downgrading to an Earlier Version of the ACNS Software

If you downgrade from the ACNS 5.4.x software to the ACNS 5.3. software, the local WebsenseEnterprise directory and other related Websense files are removed. All previously existing internal configuration files used by Websense are deleted, and all changes that were made before the downgrade are lost.

The following error message is displayed to notify you about this Websense downgrade issue:

```
WARNING:  
Websense does not support downgrade  
Hence removing /local/local1/WebsenseEnterprise  
Websense will stop working after copy ftp install
```



### Note

In previous releases, the software did not generate an error message indicating that the WebsenseEnterprise directory had been removed.

When the Content Engine reloads after the downgrade, Websense is reinstalled, and the internal configuration files are recreated. Content Engine Websense server configurations (such as **`websense-server service policy local activate`** and **`websense-server service eim activate`**) are reinitiated and they are stored in the startup configuration before the downgrade.

If you downgrade from the ACNS 5.4.x software to the ACNS 5.3. software, the startup configuration will remain intact. All previously existing configuration files are cleaned up and a fresh install is performed. All changes that were made earlier are lost. After rebooting the Content Engine, the CLI configuration is taken from the startup configuration.

If you downgrade from the ACNS 5.4.x software to the ACNS 5.2.x or 5.1.x software, the configured CLI commands and the Websense configuration files on the Content Engine remain intact.

If the local (internal) Websense server is enabled on the Content Engine and you downgrade from the ACNS 5.2.x software to either the ACNS 5.0 software or the ACNS 5.1 software, the WebsenseEnterprise directory is removed from the Content Engine and the local Websense server stops working. To avoid this problem when downgrading from the ACNS 5.2.x software or later to either the ACNS 5.1 software or the ACNS 5.0 software, follow these steps:

- 
- Step 1** Disable the local (internal) Websense server on the Content Engine.
  - Step 2** Deactivate the Websense services on the Content Engine.
  - Step 3** Install the ACNS 5.1 software or the ACNS 5.0 software downgrade image on the Content Engine.
- 

## Example of Configuring Websense Server Failover and URL Filtering

In the following example, the Content Engine is acting as the HTTP proxy for URL filtering. The Content Engine is first configured to use either the local or the remote Policy Server, and then the local Websense server services (the local EIM Server, the local User Service, and the local Network Agent) are activated on the Content Engine.

The Content Engine next is configured to use the local (internal) Websense server as its primary Websense server and an external Websense server as the secondary Websense server. If the primary Websense server is unavailable, the Content Engine sends the filtering requests to this secondary server.

After allow mode is reenabled on the Content Engine, URL filtering is enabled on the Content Engine. The Websense manager GUI is used to configure the default policy for the local and the remote Websense servers, and then the HTTP proxy is enabled on the Content Engine.

To configure Websense server failover and URL filtering, follow these steps:

- 
- Step 1** Specify whether the local or remote Websense Policy Server is to be used to activate the individual Websense services on the Content Engine.
    - To use the local Policy Server, activate the local Policy Server on the Content Engine, as follows:

```
ContentEngine(config)# websense-server service policy local activate
```

- To use a remote Policy Server, configure the necessary information about the remote Policy Server (for example, its hostname or IP address, and its port number) on the Content Engine, as follows:

```
ContentEngine(config)# websense-server service policy remote host {hostname|
IP address} [port policy-server-port]
```

where:

- *hostname* or *IP address* is the hostname or IP address of the remote Policy Server.
- The port number is optional. The default port number is 55806.

Either the local or the remote Policy Server must be running before you can activate any of the services of the local Websense server (the local EIM Server, the local Network Agent, the local eDirectory Agent, the local RADIUS Agent, and the local User Service) on the Content Engine. Local and remote Policy Server configuration are mutually exclusive.

- Step 2** Activate the local EIM Server on the Content Engine.

```
ContentEngine(config)# websense-server service eim activate
```

- Step 3** Activate the local User Service on the Content Engine.  
 ContentEngine(config)# **websense-server service user activate**
- Step 4** Activate the local Network Agent on the Content Engine.  
 ContentEngine(config)# **websense-server service network-agent activate**
- Step 5** Activate the local eDirectory Agent on the Content Engine.  
 ContentEngine(config)# **websense-server service edir-agent activate**
- Step 6** Activate the local RADIUS Agent on the Content Engine.  
 ContentEngine(config)# **websense-server service radius-agent activate**
- Step 7** Activate the local Logon Agent on the Content Engine.  
 ContentEngine(config)# **websense-server service logon-agent activate**
- Step 8** Enable all of the services of the local Websense server (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service) that have been activated on the Content Engine.  
 ContentEngine(config)# **websense-server enable**



**Note** By default, the local Websense server, which consists of the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service, is disabled on a Content Engine. If you are using the local Websense server with a cluster of standalone Content Engines, make sure that you enable the local Websense server on each standalone Content Engine (that is, enter the **websense-server enable** global configuration command on each Content Engine in the Content Engine cluster).

- Step 9** Configure the Content Engine to use the local Websense server as the primary Websense server by using the **url-filter http websense server local** global configuration command.



**Note** You can use the **url-filter http websense server** global configuration command to configure different settings (for example, the timeout, port number, and the number of connections) for the primary and secondary Websense servers. By default, the Content Engine (that is acting as the HTTP proxy) sends filtering requests to the Websense server on port 15868, waits 20 seconds for a response from the Websense server before timing out the connection, and establishes 40 persistent connections per CPU.

In this example, the Content Engine (that is acting as the HTTP proxy) sends filtering requests to the local Websense server on port 4005, waits 60 seconds for a response from the local Websense server before timing out the connection, and establishes 90 persistent connections to this local Websense server. Because the local Websense server is configured first, it is designated the primary Websense server for the Content Engine.

```
ContentEngine(config)# url-filter http websense server local port 4005 timeout 60
connections 90
```



**Note** The IP address of the local Websense server cannot be configured and is set at 127.0.0.1.

**Step 10** Configure the Content Engine to use an external Websense server as the secondary Websense server by using the **url-filter http websense server** global configuration command.

Because the local Websense server is already the primary Websense server, you must specify an external Websense server as the secondary Websense server.

In this example, the external Websense server with an IP address of 172.18.22.10 is configured as the secondary Websense server. If the local Websense server is unavailable, the Content Engine will send the requests to this secondary Websense server on port 4006, will wait up to 90 seconds for a response from this server before timing out the connection, and will establish 90 persistent connections per CPU.

```
ContentEngine(config)# url-filter http websense server 172.18.22.10 port 4006
timeout 90
```



**Note** Transitioning from primary to the secondary Websense servers is based on the timeout value (in seconds), and the maximum connections value.

If not explicitly configured, the default timeout value is 20 seconds and the default maximum connections value is 40 (per CPU). On a large WAE, the connections value is typically configured for greater than 40, otherwise the connections toggle between the two websense servers which degrades performance.

A maximum of 250 connections to a Websense server per CPU within one WAE is supported. Therefore, a WAE with two CPUs can support a maximum of 500 connections. However, the support this configuration, the WAE must be capable of handling a TPS number greater than the total configured connections.

**Step 11** By default, allow mode is enabled. To reenable allow mode, enter the following command:

```
ContentEngine(config)# no url-filter http websense allowmode enable
```

If the primary Websense server is unavailable, then the Content Engine sends the requests to the specified secondary Websense server. If both the primary and the secondary Websense servers are unavailable, then the requests are sent to allow mode.

- When allow mode is enabled, the Content Engine allows all HTTP traffic to continue through it (it proceeds with normal traffic processing) even if it fails to receive responses from the Websense server.
- When allow mode is disabled, the Content Engine blocks all HTTP traffic that is served through it if it fails to receive responses from the Websense server.

You can configure the **allowmode** option with or without the Websense server being enabled; it is independent of the Websense server configuration. The Content Engine adopts the new configuration for **allowmode** if Websense URL filtering is already being used.

**Step 12** Enable URL filtering on the Content Engine.

```
ContentEngine(config)# url-filter http websense enable
```

**Step 13** Configure the default policy using the Websense Manager GUI. This step should be performed for both the local and the remote Websense server.

- Use the Websense Manager GUI to add a Policy Server.
  - Right-click the left pane of the Websense Manager main window.
  - Choose **Add Policy Server**.

- In the displayed dialog box, enter the IP address of the Content Engine that is running the local (internal) Websense server.
- b. Connect to the Websense Policy Server that is running on the Content Engine.
- In the left pane, double-click on the Policy Server (this could be the Content Engine IP address, for example).
  - Enter the username and password, and then click **OK**.

- c. Use the Websense Manager GUI to configure a Websense policy.
  - Use the Websense Manager GUI to connect to the Websense Policy Server.
  - In the left pane, double-click **Filter Definition** and then **Policies**.
  - Choose **Global**.
  - In the right pane, click the **Edit** button.
  - In the displayed dialog box, apply such category sets as the default settings, basic settings, always block, and never block. The default policy is global, and the default category set is the default settings.



**Note** Clicking the **Save Changes** button from the Websense Enterprise Manager window does not save the Websense configuration modifications across device reboots. You need to use the **write memory** command to save the Websense configuration changes across reboots. For more information about how to use the Websense Manager GUI, go to the following website: <http://www.websense.com/content/home.aspx>.

**Step 14** Configure the HTTP proxy on the Content Engine.

```
ContentEngine(config)# http proxy incoming 8080
```

**Step 15** Display statistics for both the primary and the secondary Websense servers.

```
ContentEngine# show statistics url-filter http websense
```

## Configuring URL Filtering with a Local Websense Server

To configure the Content Engine to use the local (internal) Websense server for URL filtering, you must perform these tasks:

1. Use the local or remote Policy Server to activate the local Websense server services (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service) on the Content Engine.
2. Enable the local Websense server on the Content Engine. (By default, it is disabled.)
3. Configure the Content Engine to use the local Websense server for URL filtering of HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests). If the Network Agent is configured, other protocols can be filtered as well.

In the ACNS 5.2.1 software and later releases, you can configure up to two Websense servers for failover purposes. One of these Websense servers can be the local Websense server. The order in which you configure the Websense servers determines which server is the primary server. The first configured Websense server is automatically designated the primary Websense server, and the second configured server becomes the secondary Websense server. For a list of supported configurations, see [Table 11-8](#).

In the ACNS 5.3.1 software and later releases, you can activate any combination of the local Websense server services (which are listed in [Table 11-9](#)). If the local Policy Server is not activated on the Content Engine, you must point to a valid external Policy Server when activating the other local Websense server services (the local EIM Server, the local Network Agent, the local RADIUS Agent, the local eDirectory Agent, and the local User Service). For more information on this topic, see the “[Configuring Websense URL Filtering with External Websense Servers](#)” section on page 11-37.

The ACNS 5.0.3 to 5.1.x software has the local Websense server. Because the activation of the individual services of the Websense server in these software releases is not optional, by default the Websense server services are activated on the Content Engine as follows:

- When you upgrade to the ACNS 5.4.x software from the ACNS 5.3.x software releases, only the local Policy Server, the local EIM Server, the local eDirectory Agent, the local RADIUS agent, and the local User Service will be activated. (The local Logon Agent is not activated upon upgrade to the ACNS 5.4.1 software release).
- When you upgrade to the ACNS 5.3.x software from the ACNS 5.0.3, 5.1.x, or 5.2.x software releases, only the local Policy Server, the local EIM Server, and the local User Service will be activated. (The local eDirectory Agent and the local RADIUS Agent are not activated upon upgrade to the ACNS 5.3.1 software release).
- When you upgrade to the ACNS 5.2.1 software and later releases from the ACNS 5.0.3 or 5.1.x software releases, the following three local Websense server services are activated on the Content Engine: the local Policy Server, the local EIM Server, and the local User Service.

To configure the Content Engine to use the local (internal) Websense server for URL filtering, follow these steps:

---

**Step 1** Specify whether the local or remote Policy Server is to be used to activate the individual services of the local Websense server on the Content Engine.

- To use the local Policy Server, activate the local Policy Server on the Content Engine.

```
ContentEngine(config)# websense-server service policy local activate
```

- To use a remote Policy Server, configure the necessary information about the remote Policy Server (for example, its hostname or IP address, and its port number) on the Content Engine.

```
ContentEngine(config)# websense-server service policy remote host {hostname|
IP address} [port policy-server-port]
```

where:

- *hostname* or *IP address* is the hostname or IP address of the remote Policy Server.
- The port number is optional. The default port number is 55806.




---

**Note** Either the local or the remote Policy Server must be running before you can activate any of the services of the local Websense server (the local EIM Server, the local Network Agent, the local RADIUS Agent, the local eDirectory Agent, the local Logon Agent, and the local User Service) on the Content Engine. Local and remote Policy Server configuration are mutually exclusive.

---

**Step 2** Activate the local EIM Server on the Content Engine.

```
ContentEngine(config)# websense-server service eim activate
```

**Step 3** Activate the local User Service on the Content Engine.

```
ContentEngine(config)# websense-server service user activate
```

**Step 4** Activate the local Network Agent on the Content Engine.

```
ContentEngine(config)# websense-server service network-agent activate
```

**Step 5** Configure the settings for the local eDirectory agent, which will be running on the Content Engine, through the Websense Manager GUI (Version 5.5, or Version 6.1 or later).

**Step 6** Configure the settings for the local RADIUS Agent, which will be running on the Content Engine, through the Websense Manager GUI (Version 5.5, or Version 6.1 or later).

**Step 7** Activate the local eDirectory Agent on the Content Engine.

```
ContentEngine(config)# websense-server service edir-agent activate
```

**Step 8** Activate the local RADIUS Agent on the Content Engine.

```
ContentEngine(config)# websense-server service radius-agent activate
```

**Step 9** Enable all of the services of the local Websense server (the local EIM Server, the local Network Agent, the local Logon Agent, and the local User Service) that have been activated on the Content Engine.

```
ContentEngine(config)# websense-server enable
```



**Note** By default, the local Websense server, which consists of the local EIM Server, the local Network Agent, and the local User Service, is disabled on a Content Engine. The IP address of the local Websense server cannot be configured and is set at 127.0.0.1. If you are using the local Websense server with a cluster of standalone Content Engines, make sure that you enable the local Websense server on each standalone Content Engine (for example, enter the **websense-server enable** global configuration command on each Content Engine in the Content Engine cluster).

**Step 10** Configure the Content Engine to use the local Websense server for URL filtering of HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests). See [Table B-4](#) for the kinds of URL filtering that are supported for different protocols with a Websense server.

```
ContentEngine(config)# url-filter http websense server local [port portnumber]
[timeout seconds] [connections connections]
```

where:

- **local** specifies that the Content Engine is to use the internal Websense server for URL filtering.
- *port number* specifies the port (1–65535) on which the local Websense server is to listen for HTTP requests to filter. By default, the local Websense server listens on port 15868.
- *seconds* is the number of seconds (0–240) that the Content Engine is to wait for an HTTP response from the internal Websense server before timing out the connection. The default is 20 seconds.
- *connections* is the number of persistent connections (1–250) per CPU (the default is 40 per CPU). Use this option to configure the number of persistent connections to the internal Websense server. Do not change the default number unless you know for certain that a different value is required.

**Step 11** Enable Websense URL filtering of HTTP requests.

```
ContentEngine(config)# url-filter http websense enable
```

**Step 12** View the current Websense server configuration.

```
ContentEngine# show websense-server
```

For information about deactivating one or more of the services of the local Websense server on the Content Engine, see the [“Deactivating Local Websense Server Services on Standalone Content Engines, page 11-38.”](#)

## Configuring Websense URL Filtering with External Websense Servers

When configuring a Content Engine to use an external Websense server, you must specify an IP address and port number for that server. That specified IP address and port number must match the IP address of the external Websense server and the port that the external Websense server listens to for filtering requests. Otherwise, the Content Engine will time out all HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) and either block or allow all HTTP traffic based on the **allowmode** option configuration. By default, allow mode is enabled on the Content Engine. When allow mode is enabled, the Content Engine is permitted to fulfill an HTTP request from a client if the external Websense server does not respond. If allow mode has been disabled, use the **url-filter http websense allowmode enable** command to reenabte it.

In the ACNS 5.2.1 software and later releases, you can configure up to two Websense servers for failover purposes. The order in which you configure the Websense servers determines which server is the primary server. The first configured Websense server is automatically designated the primary Websense server, and the second configured server becomes the secondary Websense server. For a list of supported Websense server configurations, see [Table 11-8](#).

To configure a standalone Content Engine to use an external Websense server for URL filtering, follow these steps:

- 
- Step 1** Specify the necessary information about the external Websense server by using the **url-filter http websense server** global configuration command.

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

where:

- *hostname* is the hostname of the external Websense server.
- *IP address* is the IP address of the external Websense server.
- *portnum* is the port number (1–65535) of the external Websense server to which the Content Engine is to send HTTP requests. The default is port 15868.
- *seconds* is the number of seconds (0–240) that the Content Engine is to wait for an HTTP response from the external Websense server before timing out the connection. The default is 20 seconds.
- *connections* is the number of persistent connections (1–250) per CPU (the default is 40 per CPU). Use this option to configure the number of persistent connections to the external Websense server. Do not change the default number unless you know for certain that a different value is required.

The following example shows how to configure a standalone Content Engine to point to an external Websense server that has the IP address 172.18.22.10 and is running on Host A. The Content Engine is configured to send requests to this external Websense server on port 4006, and to wait up to 90 seconds for a response from this server before timing out the connection.

```
ContentEngine(config)# url-filter http websense server 172.18.22.10 port 4006 timeout 90
```



**Note**

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

---

**Step 2** If you want to configure a secondary Websense server for failover purposes, take one of the following actions:

- To configure the local (internal) Websense server as the secondary Websense server, use the **url-filter http websense server local** global configuration command. For more information about configuring the local Websense server, see the “[Configuring URL Filtering with a Local Websense Server](#)” section on page 11-34.
- To configure an external Websense server that is running on a different host (Host B) than the primary Websense server (Host A), enter another **url-filter http websense server** command. This time, the command should specify the parameters for the secondary Websense server (for example, the IP address, port number, timeout, and number of connections of the Websense server running on Host B.)



**Note** Transitioning from primary to the secondary Websense servers is based on the timeout value (in seconds), and the maximum connections value.

If not explicitly configured, the default timeout value is 20 seconds and the default maximum connections value is 40 (per CPU). On a large WAE, the connections value is typically configured for greater than 40, otherwise the connections toggle between the two websense servers which degrades performance.

A maximum of 250 connections to a Websense server per CPU within one WAE is supported. Therefore, a WAE with two CPUs can support a maximum of 500 connections. However, to support this configuration, the WAE must be capable of handling a TPS number greater than the total configured connections.

**Step 3** Enable Websense as the current URL filtering scheme for HTTP on this Content Engine.

```
ContentEngine(config)# url-filter http websense enable
```



**Note** For information about configuring an external Websense server, go to the following website: <http://www.websense.com/content/home.aspx>.

**Step 4** View the current Websense server configuration.

```
ContentEngine# show websense-server
```

## Deactivating Local Websense Server Services on Standalone Content Engines

To deactivate one or more of the services of the local Websense server (the local EIM Server, the local Network Agent, the local RADIUS Agent, the local eDirectory Agent, the local User Service, or the local Policy Server) on a standalone Content Engine, follow these steps:

**Step 1** Check whether the local Websense server is currently enabled on the Content Engine.

```
ContentEngine# show websense-server
```

**Step 2** Disable the local Websense server on the Content Engine, if it is currently enabled.

```
ContentEngine(config)# no websense-server enable
```

- Step 3** Deactivate a particular service of the local Websense server by using the **no** form of the **websense-server service** global configuration command.

For example, use the **no websense-server service network-agent activate** command to deactivate the local Network Agent on the Content Engine:

```
ContentEngine(config)# no websense-server service eim activate
ContentEngine(config)# no websense-server service user activate
ContentEngine(config)# no websense-server service network-agent activate
ContentEngine(config)# no websense-server service edir-agent activate
ContentEngine(config)# no websense-server service radius-agent activate
ContentEngine(config)# no websense-server service logon-agent activate
```

There is no required order in which you should deactivate the local EIM Server, the local Network Agent, the local eDirectory Agent, the local RADIUS Agent, the local Logon Agent, or the local User Service on the Content Engine. However, if the Policy Server is running on the Content Engine (that is, the local Policy Server is being used instead of a remote Policy Server), the Policy Server should be the last local Websense service to be deactivated (using the **no websense-server service policy activate** command).

In contrast, if you are using a remote Policy Server, make sure that it is running before you attempt to deactivate the local EIM Server, the local Network Agent, the local eDirectory Agent, the local RADIUS Agent, or the local User Service (using the **no websense-server service service-name activate** command) on the Content Engine.

- Step 4** Deactivate the local Policy Server on the Content Engine, if it is being used.

```
ContentEngine(config)# no websense-server service policy
```

- Step 5** Unconfigure the remote Policy Server on the Content Engine, if it is being used.

```
ContentEngine(config)# no websense-server service policy remote host
```

## Saving Websense Configuration Files

In the ACNS 5.2.1 software and later releases, the **write memory** command saves modified Websense configuration files (the `eimserver.ini`, `config.xml`, and `websense.ini` files and the Blockpages directory) across disk reconfiguration and the ACNS software release upgrades.

You must execute the **write memory** command in order to save the most recent configuration modifications, including `websense.ini` file modifications and Websense URL filtering configuration changes. The **write memory** command enables the changes made from the external Websense Manager GUI to be saved across disk reconfiguration and upgrades (which might erase disk content).

The Websense configurations from the last use of the **write memory** command are retained under the following situations:

- If the **write memory** command is not used before a reboot with a disk reconfiguration or an ACNS software upgrade that erases disk content
- If you are using the **reload** command and did not answer **yes** when asked if you wanted to save the configurations at the reload prompt

If the **write memory** command has never been used before, then default configurations will be applied when the content in the `/local1/WebsenseEnterprise/EIM` directory on the Content Engine is erased.

## Viewing Websense URL Filtering Statistics

To display the status of all HTTP URL filtering schemes presently configured on the standalone Content Engine, enter the **show url-filter http** EXEC command. Use the **show statistics url-filter http websense** EXEC command To display the request-reply statistics for the communication between the Content Engine and the Websense server, enter the **show statistics url-filter http websense** EXEC command. These statistics show the number of requests sent, replies received, pages blocked, pages allowed, and failure cases. More detailed URL filtering statistics are available on the Websense server.

The statistics shown can be cleared using the **clear statistics url-filter http websense** and **clear statistics all** EXEC commands. All the statistics counters are then reset to 0.

## Configuring URL Filtering with SmartFilter Software

SmartFilter software running on standalone Content Engines provides employee Internet management (EIM) functionality when used with proxy servers, firewalls, and caching appliances. The SmartFilter filtering capability is available as an add-on service on a Content Engine that is running the ACNS 5.x software. The SmartFilter add-on service is licensed directly through Cisco.

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

To configure this SmartFilter add-on service, you use an end user management tool called the sfadmin console, and a management server tool called the sfadmin server. You use the sfadmin console to configure the SmartFilter product and then store the configuration on the sfadmin server. The sfadmin server propagates this configuration to the end client Content Engines, to be used by the SmartFilter software that is running on the Content Engines. To enable SmartFilter URL filtering on a standalone Content Engine, use the **url-filter http smartfilter enable** global configuration command. To use SmartFilter URL filtering with a cluster of standalone Content Engines, make sure to enter the **url-filter http smartfilter enable** command on each Content Engine in the cluster to ensure that all traffic is filtered.



### Note

The ACNS 5.2.1, 5.3.x, 5.4.1, 5.5.1, and 5.5.3 software releases support SmartFilter software Version 4.0. SmartFilter software Version 4.1 is supported in the ACNS 5.4.3 release. With the SmartFilter software Version 4.1 support, the agent listens directly on port 9014 for block page requests. See [Table B-4](#) for a complete list of protocols that are supported with SmartFilter. See the Release Notes for the most current SmartFilter software version support information for your ACNS release.

When configuring URL filtering with Smartfilter software, remember the following important points:

- When you upgrade or downgrade the Content Engine to a different release of the ACNS software, if there is a difference in the SmartFilter plug-in version, the SmartFilter database and configuration files are deleted and default configurations are loaded. This change occurs because the configuration details might be changed with each new version of the SmartFilter software. After each upgrade or downgrade of the SmartFilter plug-in, a fresh database has to be downloaded from the SmartFilter Administration Console to the Content Engine.
- If the Content Engine is deployed in a natted environment, you must use the **external-ip external-ip-address** global configuration command to configure the Content Engine's external IP address, otherwise the Advanced block page feature may not work properly.
- The Smartfilter software and the cache process on the Content Engine will be restarted in the following situations:

- If you configure the Content Engine's external IP address when the Smartfilter software is running on the Content Engine.
- If the Content Engine's external IP address is not configured and you change the Content Engine's interface IP address when the Smartfilter software is running on the Content Engine.
- Port 9014 is reserved for Smartfilter Advanced block pages.

## About the SmartFilter Control List

A SmartFilter Control List categorizes 2 million websites into content groups. There are 30 predefined SmartFilter Control List categories that encompass a wide variety of material. Some categories are focused on reducing legal liability of a company. These 30 categories are set to Deny in the default SmartFilter software policy. Some categories contain such sites as MP3 sites (sites with content that consumes excessive bandwidth). The remainder of these 30 categories are considered unproductive or inappropriate for business or educational environments.

The SmartFilter software also provides ten user-defined categories that allow you to further tailor access by defining and filtering sites that are not included in the SmartFilter Control List. Additionally, you can exempt any site that you would like specific groups or individuals to access quickly and easily. You can use the SmartFilter Administration Console to define a SmartFilter Control List download schedule. The Download Setup window tracks the download site, your username, and your password. If you do not download an updated SmartFilter Control List at least monthly, the SmartFilter software considers the Control List expired, and invokes the action that you specified in the SmartFilter License window.


**Note**

For ACNS 5.5.9 software and later, filtering of the SmartFilter control list is supported when the control list size exceeds 200 MB.

## About the Temporary User Override Feature

In the ACNS 5.4.1 software release, the temporary user override was a new feature that is available with the SmartFilter software Version 4.1. This feature allows certain users to override the filtering process that is currently being applied to their user group. You must configure this feature through the SmartFilter Administrator Console.

To use the temporary user override feature, follow these steps:

- 
- Step 1** Install the SmartFilter authentication server software on the machine that is running the SmartFilter Administrator Console or on a different machine.



**Note** To obtain a copy of the Smartfilter authentication server software, go to the following website: <http://www.securecomputing.com>.

- Step 2** From the SmartFilter Administrator Console, add the authentication server.
- Step 3** From the SmartFilter Administrator Console, add the users the authentication server.
- Step 4** Send the changes to the authentication server.
- Step 5** From the SmartFilter Administrator Console, select the Content Engine and add the configured authentication server to the Content Engine's list of authentication servers.
- Step 6** From the SmartFilter Administrator Console, add the users who should be allowed to override the filtering process in the overrides for the Content Engine.
- Step 7** Send the changes to the Content Engine.


**Note**

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

# Configuring Content Engines to Bypass URL Filtering for Specific HTTP and HTTPS Requests

In the ACNS 5.2.3 software and later releases, you can configure a Content Engine to bypass URL filtering for certain HTTP and HTTPS requests. This feature is supported for local list URL filtering (good and bad site lists), as well as Websense, SmartFilter, or N2H2 URL filtering.

For example, if you enable local URL filtering on the Content Engine and enable the bad sites deny feature (for example, the badfile.txt file contains the URLs that should be blocked), and the **rule no-url-filtering** action is a hit (a match), the Content Engine bypasses the URL filtering for that particular request; otherwise, it proceeds with URL filtering and blocks the URL request.

To configure this feature on a standalone Content Engine, use the **rule action no-url-filtering** global configuration command. For more information on this topic, see the [“Example of no-url-filtering Action” section on page 13-14](#).

## Displaying the Current URL Filtering Configurations

To display the URL filtering configurations for a standalone Content Engine, use the **show url-filter EXEC** commands:

```
ContentEngine# show url-filter http
ContentEngine# show url-filter rtsp
ContentEngine# show url-filter wmt
```



### Caution

If the size of the local list file becomes too large, it can adversely effect proxy performance, because the local list file is loaded into memory when local list filtering is enabled. If the file size is larger than 5 MB, a warning message appears, but the ACNS software does not enforce size limits for the local list file. It is your responsibility to track the local list file size and ensure that it does not become so large that it degrades performance.

## Displaying URL Filtering Statistics

To display statistics for the various URL filtering schemes that are configured on a standalone Content Engine, use the **show statistics url-filter EXEC** commands.

```
ContentEngine# show statistics url-filter ?
  http  Display URL-filter for http and mms over http statistics
  rtsp  Display URL-filter for rtsp statistics for real proxy, real server and
        cisco streaming engine
  wmt   Display URL-filter for wmt statistics for rtsp requests
```

To display local list URL filtering statistics for WMT requests (RTSP requests from Windows Media 9 players), enter the **show statistics url-filter wmt local-list EXEC** command.



**Note** For WMT requests, local list files is the only supported URL filtering scheme.

As the sample command output shows, the number of allowed WMT requests, blocked WMT requests, and WMT requests not filtered by the local list URL filtering are displayed:

```
ContentEngine# show statistics url-filter wmt local-list
```

```
Local List URL filtering statistics:
    Requests Allowed = 25
    Requests Blocked = 30
    Requests not filtered = 5
```

To display RealMedia request statistics for a standalone Content Engine, enter the **show statistics url-filter rtsp local-list** EXEC command. RealMedia requests are handled by the RealProxy server that is running on the standalone Content Engine.

```
ContentEngine# show statistics url-filter rtsp?
    local-list  Display local-list URL-filter statistics
```



**Note** For RealMedia requests, local list files is the only supported URL filtering scheme. For registered Content Engines (that is Content Engines that are registered with a Content Distribution Manager as opposed to standalone Content Engines that are initially not registered with a Content Distribution Manager if there is one in the ACNS network), the command output from the **show statistics url-filter rtsp local-list** EXEC command will also include statistics that the RealSubscriber and Cisco Streaming Engine have served to clients if these two backend RTSP servers have been enabled on the registered Content Engine.

As the sample command output shows, the number of allowed requests, blocked requests, and the number of requests not filtered by the local list URL filtering are displayed:

```
ContentEngine# show statistics url-filter rtsp local-list
Local List URL filtering statistics:
    Requests Allowed = 15
    Requests Blocked = 10
    Requests not filtered = 2
```

To display URL filtering statistics for HTTP requests, enter the **show statistics url-filter http** EXEC command. With URL filtering for HTTP requests, local list files and URL filtering through third-party software (for example, N2H2 and Websense software) are supported.

```
ContentEngine# show statistics url-filter http ?
    local-list  Display local-list URL-filter statistics
    N2H2        Display N2H2 URL-filter statistics
    websense    Display websense URL-filter statistics
```

## Clearing URL Filtering Statistics

To clear URL filtering statistics on standalone Content Engines, use the **clear statistics url-filter** EXEC commands.

```
ContentEngine# clear statistics url-filter ?
    http  Clear URL-filter for http statistics
    rtsp  Clear URL-filter for rtsp statistics
    wmt   Clear URL-filter for wmt statistics
```

For example, clear the WMT URL filtering statistics on a standalone Content Engine, as follows:

```
ContentEngine# clear statistics url-filter wmt local-list
```