



# Configuring URL Filtering on Standalone Content Engines

This chapter describes the different types of URL filtering that are supported with standalone Content Engines that are running ACNS 5.x software, and describes how to configure URL filtering on standalone Content Engines.

This chapter contains the following sections:

- [URL Filtering Overview, page 10-1](#)
- [Displaying the Current URL Filtering Configurations, page 10-3](#)
- [URL Filtering with Local List Files, page 10-4](#)
- [URL Filtering with an N2H2 Server, page 10-8](#)
- [URL Filtering with Websense Software, page 10-11](#)
- [URL Filtering with SmartFilter Software, page 10-24](#)



**Note**

For complete syntax and usage information for the CLI commands used in this chapter, refer to the *Cisco ACNS Software Command Reference, Release 5.2* 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), refer to the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.2*.

## URL Filtering Overview

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 10-1](#) 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 10-1** URL Filtering Mechanisms 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="#">URL Filtering with Local List Files</a> ” section on page 10-4.
N2H2 external servers	Direct client requests for content to an external N2H2 server for URL filtering. See the “ <a href="#">URL Filtering with an N2H2 Server</a> ” section on page 10-8.
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="#">URL Filtering with Websense Software</a> ” section on page 10-11.
SmartFilter plug-in	Direct client requests for content to the SmartFilter plug-in for URL filtering. See the “ <a href="#">URL Filtering with SmartFilter Software</a> ” section on page 10-24.

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
```

## RADIUS and URL Filtering

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

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

The Filter-Id attribute is defined as either No-Web-Blocking or Yes-Web-Blocking. 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 16-6.

## 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
```

In this example, the **show url-filter http** command is used to display the status of all URL filtering schemes for HTTP requests that are currently configured on the standalone Content Engine.

```
ContentEngine# show url-filter http
URL filtering is set to use bad-list

Local list configurations
=====
Good-list file name :
Bad-list file name : /local1/url-filter/badlist.http
Custom message directory :

Websense server configuration
=====
Websense server IP   : 172.16.193.165
Websense server port : 15868
Websense server timeout: 20 (in seconds)
Websense server connections : 40
Websense allow mode is ENABLED

N2H2 server configuration
=====
N2H2 server IP      : 172.16.193.165
N2H2 server port    : 4005
N2H2 server timeout : 5 (in seconds)
N2H2 allow mode is ENABLED
ContentEngine#
```

## URL Filtering with Local List Files

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 streaming media protocols such as MMS and RTSP. This type of URL filtering is referred to as “local list URL filtering.”



**Tip**

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

The local list file for each protocol should not contain URLs that belong to other protocols. For instance, the HTTP local list file should contain only HTTP, HTTPS, or FTP URLs, and the WMT local list file should contain only MMS 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.

## Configuring Local List URL Filtering on Standalone Content Engines

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)
- Requests over RTSP
- WMT requests

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

To use the Content Engine CLI to configure local list URL filtering on a standalone Content Engine, use the **url-filter** global configuration commands.

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

Local list URL filtering is the only supported filtering mechanism for requests over RTSP and WMT requests; the N2H2, SmartFilter, and Websense filtering mechanisms are not supported for these types of requests. For requests over HTTP, the local list URL filtering mechanism as well as N2H2, SmartFilter, and Websense is supported. See [Table B-4](#) for a list of the kinds of URL filtering that are supported for the different protocols.

Table 10-2 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, and HTTPS-over-HTTP requests).

**Table 10-2** 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 10-3 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for requests over RTSP.

**Table 10-3** 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 <i>filename</i></b>	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 <i>filename</i></b>	Specifies the filename of the RTSP good site list.

Table 10-4 describes the Content Engine CLI commands for configuring a standalone Content Engine to use local list URL filtering for Windows Media Technologies (WMT) requests (MMS requests over UDP [MMSU], MMS requests over TCP [MMS], and MMS requests over HTTP).

**Table 10-4** 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 <i>filename</i></b>	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 <i>filename</i></b>	Specifies the filename of the WMT good site list.

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** Use the `url-filter http bad-sites-deny` global configuration command to point to the bad URL list.

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

**Step 4** Use the `url-filter http bad-sites-deny enable` global configuration command to actively deny the URLs.

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

---

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** Use the `url-filter http good-sites-allow file` global configuration command to point to the *goodurl.lst* file.

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

**Step 4** Use the `url-filter http good-sites-allow enable` global configuration command to actively permit only the good URLs.

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

---

## 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}**

where:

- **http** reloads the new local lists for HTTP requests (HTTP, FTP-over-HTTP, and HTTPS-over-HTTP requests).
- **rtsp** reloads the local lists for requests over RTSP.
- **wmt** reloads the local lists for WMT requests (MMSU, MMST, and MMS-over-HTTP requests).

The following 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 your having to reenter the **url-filter http custom-message** command.

In the following example, a *block.html* file displays the custom message

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

when the standalone Content Engine intercepts a request to the blocked site.

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.

## URL Filtering with an N2H2 Server

N2H2 is a globally deployed URL-filtering solution that can filter HTTP, FTP, or HTTPS requests based on destination host name, destination IP address, and user name and password. It 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 <http://www.n2h2.com> for further information on N2H2 filtering products.



**Note**

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

N2H2 supports three filtering methods. [Table 10-5](#) lists the N2H2 features supported by the Content Engine. One N2H2 server can support multiple Content Engines simultaneously.

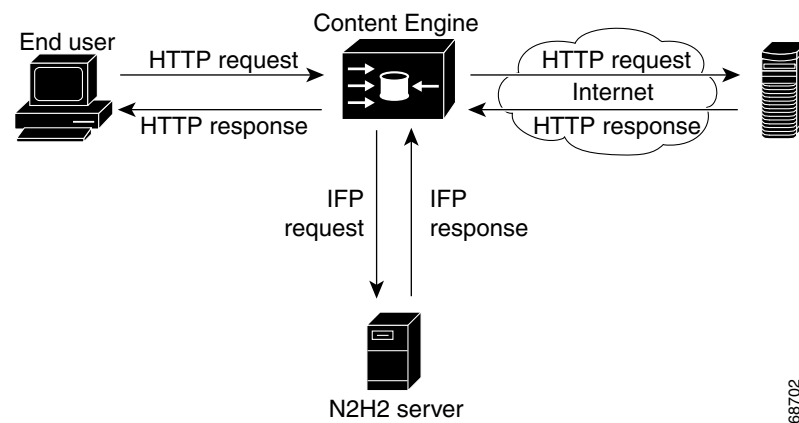
**Table 10-5 Supported N2H2 Features**

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.

## Configuring Standalone Content Engines for N2H2 URL Filtering

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 10-1](#).) 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 10-1** N2H2 Filtering



68702



**Note**

URL filtering using an N2H2 server is applied to HTTP traffic (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) before the Rules Template mechanism is applied, 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** Use the `show url-filter http EXEC` command to display the URL filtering schemes that are currently enabled on this Content Engine for requests over 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** Use the **url-filter http N2H2 server** global configuration command to configure the Content Engine to use an external N2H2 server for URL filtering.

- 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]]

where:

- *host name* is the host name 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 (1–20) 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 15 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 15
```

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** Use the **url-filter http N2H2 enable** global configuration command to enable the N2H2 URL filtering scheme on this Content Engine.

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

**Step 5** Use the **url-filter http N2H2 allowmode enable** global command to 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 default, **allowmode** is enabled.

- 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.

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** Use the **show statistics url-filter http N2H2 EXEC** command to display the request-reply statistics for the communication between the Content Engine and the N2H2 server.

These statistics show the number of requests sent, replies received, pages blocked, pages allowed, and failure cases. More detailed URL filtering statistics are available on the N2H2 server.

The statistics shown can be cleared using the **clear statistics url-filter http N2H2** and **clear statistics all EXEC** commands. The **clear statistics url-filter http N2H2 EXEC** command resets the statistics counters for the N2H2 server. All the statistics counters are reset to 0.



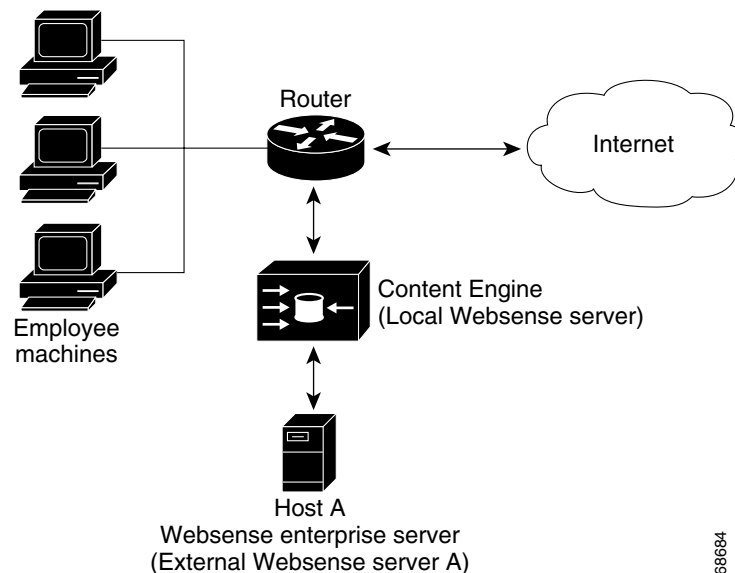
**Note**

Go to <http://www.n2h2.com> for further information on N2H2 filtering configuration and policies.

## URL Filtering with Websense Software

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 10-2](#) 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 10-2** 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 ACNS software, Release 5.1.x and earlier, only one Websense server is supported. In ACNS software, Release 5.2 or later, up to two Websense servers are supported. For more information on this topic, see the next section, “[About Websense Server Failover](#).”

## About Websense Server Failover

In ACNS software, Release 5.2, the Websense server failover feature was added. This feature allows you to configure a Content Engine to use up to two Websense servers for failover purposes (one primary and one secondary server) for URL filtering. [Table 10-6](#) lists the supported Websense server failover configurations.

**Table 10-6** Supported Websense Server Failover Configurations

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

The order in which you configure 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 [“Configuring Websense Server Failover and URL Filtering”](#) section on page 10-16.

## About Websense Services

In Websense 5.2 software, three services (the employee Internet management [EIM] server, network agent service, and user service) replace the single Websense service (the local Websense server) that was supported in Websense Version 5.0.1. The term “local Websense server” is still used to refer collectively to these three Websense processes that are running internally on the Content Engine. These Websense processes are also called “services.”

With Websense 5.2 software, you can use a local or remote Websense policy server to activate the local EIM server, the local network agent, and the local user service individually on a Content Engine. See [Table 10-7](#).

**Table 10-7** Services of the Local Websense 5.2 Server

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, 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, 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 user service	Enables URL filtering based on user-based or group-based policies. If you are using a user service and you want to configure user-based or group-based URL filtering using a Windows NT directory, then you must use the external user service on a Windows machine.

[Table 10-8](#) lists the Content Engine CLI commands that were added in ACNS 5.2 software in order to support the integration of Websense 5.2 software.

**Table 10-8** Websense Server 5.2 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> [ <b>host</b> <i>remote-policy-server IP-address</i> ] [ <b>port</b> <i>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.

**Note**

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

## Configuring Standalone Content Engines for Websense URL Filtering

In ACNS software, Release 5.2, 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 “[Configuring Websense Server Failover and URL Filtering](#)” section on page 10-16.
- To use only the local Websense server, see the “[Configuring URL Filtering with a Local Websense Server](#)” section on page 10-19.
- To use only an external Websense server, see the “[Configuring Websense URL Filtering with External Websense Servers](#)” section on page 10-21.

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. Use the **show url-filter http** EXEC command 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). See [Table B-4](#) for the kinds of URL filtering that are supported for different protocols with a Websense server.

**Note**

Cisco ACNS 5.2 software supports Websense server Version 5.2 on all Cisco Content Engine platforms. For more detailed information about configuring the Websense software, go to the following website: <http://www.websense.com>.

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

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/downloads>

## 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 10-9](#).

**Table 10-9** 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 <i>websense.ini</i> file for this port, and we recommend that you do not modify this default.	55806

You can configure the first three ports that are listed in [Table 10-9](#) 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, refer to the *Cisco ACNS Software Command Reference, Release 5.2* publication.

## Configuring Websense Server Failover and URL Filtering

In the following scenario, the Content Engine is acting as the HTTP proxy for URL filtering. First, the Content Engine is 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.

Next, the Content Engine 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. Finally, 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.

---

**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, use the **websense-server service policy local activate** global configuration command to activate the local policy server on the Content Engine.

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

- To use a remote policy server, use the **websense-server service policy remote host** global configuration command to configure the necessary information about the remote policy server (for example, its host name 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 host name 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, and the local user service) on the Content Engine. Local and remote policy server configuration are mutually exclusive.

**Step 2** Use the **websense-server service eim activate** global configuration command to activate the local EIM server on the Content Engine.

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

**Step 3** Use the **websense-server service user activate** global configuration command to activate the local user service on the Content Engine.

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

**Step 4** Use the **websense-server service network-agent activate** global configuration command to activate the local network agent on the Content Engine.

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

**Step 5** Use the **websense-server enable** global configuration command to enable all of the services of the local Websense server (the local EIM server, the local network agent, and the local user service) that have been activated on the Content Engine. 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.

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

**Note**

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 6** Configure the Content Engine to use the local Websense server as the primary Websense server (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 7** Configure the Content Engine to use an external Websense server as the secondary Websense server (using the **url-filter http websense server** global configuration command).

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

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

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

- Step 8** 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 9** Enable URL filtering on the Content Engine.

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

**Step 10** 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.
- 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**.
- 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** For more information about how to use the Websense Manager GUI, go to the following website: <http://www.websense.com>.

**Step 11** Configure the HTTP proxy on the Content Engine.

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

**Step 12** Use the **show statistics url-filter http websense EXEC** command to display statistics for both the primary and the secondary Websense servers.

## Configuring URL Filtering with a Local Websense Server

In order 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, and the local user service) on the Content Engine.

In ACNS software, Release 5.2 or later, you can now activate one or more Websense server services individually. [Table 10-7](#) lists the Websense server services.

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 ACNS software, Release 5.2 or later, 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, whereas the second configured server becomes the secondary Websense server. For a list of supported configurations, see [Table 10-6](#).

In ACNS software, Release 5.2 or later, you can activate any combination of the local Websense server services (which are listed in [Table 10-7](#)). 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 three local Websense server services (the local EIM server, the local network 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 10-21](#).

ACNS software, Release 5.0.3 to 5.1.x, has the local Websense server. Because the activation of the individual services of the Websense server in these software releases is not optional, when you upgrade to ACNS software, Release 5.2 from ACNS software, Release 5.0.3 to Release 5.1.x, by default 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, use the **websense-server service policy local activate** global configuration command to activate the local policy server on the Content Engine.

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

- To use a remote policy server, use the **websense-server service policy remote host** global configuration command to configure the necessary information about the remote policy server (for example, its host name 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 host name 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, and the local user service) on the Content Engine. Local and remote policy server configuration are mutually exclusive.

**Step 2** Use the **websense-server service eim activate** global configuration command to activate the local EIM server on the Content Engine.

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

**Step 3** Use the **websense-server service user activate** global configuration command to activate the local user service on the Content Engine.

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

**Step 4** Use the **websense-server service network-agent activate** global configuration command to activate the local network agent on the Content Engine.

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

**Step 5** Use the **websense-server enable** global configuration command to enable all of the services of the local Websense server (the local EIM server, the local network agent, and the local user service) that have been activated on the Content Engine. 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.

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



**Note** 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 6** Use the **url-filter http websense server local** global configuration command to 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 7** Use the **url-filter http websense enable** global configuration command to enable Websense URL filtering of HTTP requests.

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

- Step 8** Use the **show websense-server EXEC** command to view the current Websense server configuration.

---

For information about deactivating one or more of the services of the local Websense server on the Content Engine, see the “[Configuring Websense URL Filtering with External Websense Servers, page 10-21.](#)”

## 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 reenab it.

In ACNS software, Release 5.2 or later, 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, whereas the second configured server becomes the secondary Websense server. For a list of supported Websense server configurations, see [Table 10-6](#).

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

- Step 1** Use the **url-filter http websense server** global configuration command to specify the necessary information about the external Websense server.

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

where:

- *host name* is the host name 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 10-19](#).
  - To configure an external Websense server that is running on a different host (Host B) than the primary Websense server (Host A), enter another **url-filter http websense server** command. This time, the command should specify the parameters for the secondary Websense server (for example, the IP address, port number, timeout, and number of connections of the Websense server running on Host B.)

- Step 3** Use the **url-filter http websense enable** global configuration command to enable Websense as the current URL filtering scheme for HTTP on this Content Engine.

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



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

- Step 4** Use the **show websense-server EXEC** command to view the current Websense server configuration.

## 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 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 by using the **show websense-server EXEC** command.
- Step 2** If the local Websense server is currently enabled on the Content Engine, use the **no websense-server enable** global configuration command to disable it.
- Step 3** Use the **no** form of the **websense-server service** global configuration command to deactivate a particular service of the local Websense server. For example, use the **no websense-server service network-agent activate** 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
```

There is no required order in which you should deactivate the local EIM server, the local network 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, or the local user service (using the **no websense-server service service-name activate** command) on the Content Engine.

- Step 4** If the local policy server is being used, use the **no websense-server service policy** global configuration command to deactivate it on the Content Engine. If a remote policy server is being used, use the **no websense-server service policy remote host** global configuration command to unconfigure it on the Content Engine.

## Saving Websense Configuration Files

In ACNS software, Release 5.2 or later, 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 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

Use the **show url-filter http EXEC** command to display the status of all HTTP URL filtering schemes presently configured on the standalone Content Engine. 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. 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.

## 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 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. Use the **url-filter http smartfilter enable** global configuration command to enable SmartFilter URL filtering on a standalone Content Engine. 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**

---

Cisco ACNS 5.2 software supports SmartFilter software Version 4.0. See [Table B-4](#) for a complete list of protocols that are supported with SmartFilter.

---

## 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.

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

---

