



Configuring Content Filtering Services

You can configure various content filtering services for the Content Engine, such as ICAP services, rules, and URL filtering. The following sections describe how to define and configure these various features:

- [Configuring ICAP Services, page 10-1](#)
- [Configuring Service Rules, page 10-8](#)
- [Configuring URL Filtering, page 10-17](#)

Configuring ICAP Services

The Internet Content Adaptation Protocol (ICAP) is an open-standards protocol that can be used for content adaptation, typically at the network edge. ICAP provides a common standard for communication between edge devices and ICAP services such as content filtering and virus scanning provided by third-party vendors. ICAP support on Content Engines enables system resources to be used for other critical applications while using the services offered by third-party vendors for content adaptation.

There are various vector points (decision points) related to ICAP that enable modification of HTTP requests and responses when they are processed by the Content Engine. You can define one or more ICAP services and associate one or more vector points that the service is able to support. An ICAP service is a collection of attributes and ICAP servers. You can configure a maximum of ten ICAP services per Content Engine, with an upper limit of five ICAP servers per ICAP service. Also, you can choose to apply ICAP services to all HTTP requests processed by the Content Engine or apply ICAP processing only to requests that match the Rules Template.



Note

The maximum file size that is supported in the ACNS software in pass-through mode is 2 GB. Files that exceed this size limit are not supported for ICAP processing.



Note

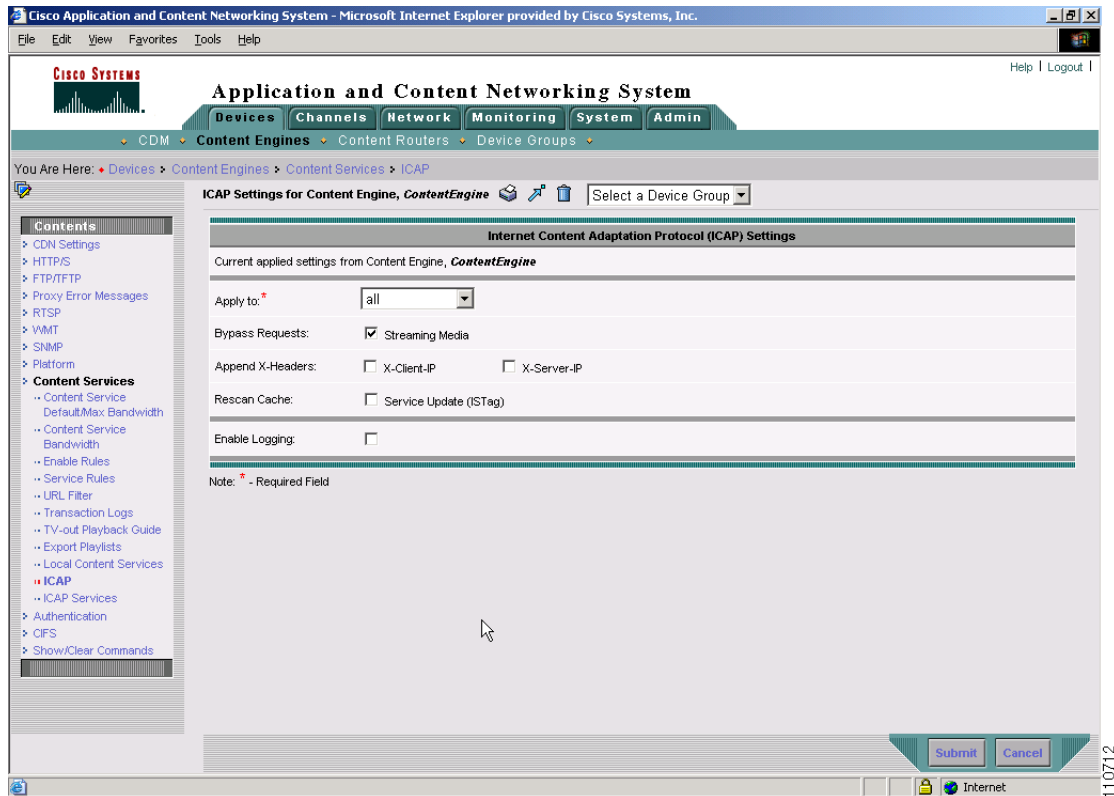
To configure ICAP services using the CLI, refer to the *Cisco ACNS Software Caching and Streaming Configuration Guide, Release 5.1*.

Configuring ICAP Settings

To configure ICAP settings for the Content Engine, follow these steps:

- Step 1** Choose **Devices > Content Engines**. The Content Engines window appears.
- Step 2** Click the **Edit** icon next to the desired Content Engine. The Modifying Content Engine window appears.
- Step 3** In the Contents pane, choose **Content Services > ICAP**. The ICAP Settings for Content Engine window appears.

Figure 10-1 ICAP Settings Window



- Step 4** From the Apply drop-down list, choose the type of ICAP processing that you wish to apply to requests: The **all** option applies ICAP processing to all HTTP requests. The **rules-template** option applies ICAP processing only to requests that match the **use-icap-service** rule defined in the Rules Template.
- Step 5** The **Streaming Media** check box for bypass requests is checked by default, meaning that ICAP processing is bypassed for streaming media requests (that is, requests from Windows, Real Media, and QuickTime media players) that enter the Content Engine.
- Step 6** To add an X-client-IP header for ICAP processing of HTTP requests, check the **X-Client-IP** check box.
- Step 7** To add an X-server-IP header for ICAP processing, check the **X-Server-IP** check box.
- Step 8** To rescan the cached objects, check the **Service Update (ISTag)** check box. Changing or updating the ISTag is a method of informing the ICAP client that any of the previously cached responses on the client should not be used.

When a file is scanned by an ICAP virus scanner and no virus is detected, the object is stored in the cache. However, an update to the ICAP virus scanner might be able to capture a new virus in the cached object. In such cases, you can use the option of rescanning cached objects when the ISTag changes.

- Step 9** Check the **Enable Logging** check box if you wish to configure and enable logging for ICAP exchanges between ICAP servers and Content Engines. This check box must be checked if you want to create logs in ICAP standard logging format.
- Step 10** Click **Submit** to save your settings. A “Click Submit to Save” message appears in red next to the current settings line when there are pending changes to be saved. To revert to the previously configured window settings, click **Reset**. The Reset button appears only when you have applied default or group settings to change the current device settings but the settings have not yet been submitted.
- Step 11** To delete the configured settings for the device, click the **Remove Settings** icon in the taskbar to delete the settings. This icon appears only if you have configured the settings for the Content Engine.
- Step 12** To restore the factory default settings to the device, click the **Apply Defaults** icon in the taskbar.
- Step 13** To override the device group settings applied to the device with the factory default settings, click the **Override Group Settings with Defaults** icon in the taskbar. This icon appears only if you have applied the device group settings to the Content Engine.
- Step 14** When settings have been applied from device groups with which the device is associated, click the **Override Group Settings** icon in the taskbar to override the device group settings and configure the device settings. This icon appears only if you have applied the device group settings to the Content Engine.
- Step 15** When a device is associated with one or many device groups that have been configured with ICAP settings, choose the device group name from the drop-down list that appears in the taskbar if you want to apply settings from a different device group to this device.
-

Configuring ICAP Services

ACNS 5.1 software, supports the following three vector points:

- Request modification-precache

This vector point applies to requests from clients. The ICAP server specifies whether to terminate the connection, send a modified error response to the client, search the cache using the requested URL, use a modified URL before looking up the cache, or modify the request headers or bodies, in the case of a cache miss.

- Request modification-postcache

This vector point applies to requests only in the case of a cache miss and before the request is to be forwarded to the origin server for content retrieval. The ICAP server determines whether to terminate the client connection, send a modified error response to the client, send the request to the origin server using the client-specified URL or an alternative URL, or modify the request headers or bodies.

- Response modification-precache

This vector point applies to responses that are received from the origin server. The ICAP server specifies whether to return the response to the client, modify the response headers or bodies before sending them to the client, or cache the response using the same or an alternative URL.

ICAP servers configured with various vector points (especially request modification-precache) may become overloaded with HTTP requests because all requests pass through this point. We therefore recommend that you use a cluster of ICAP servers to load-balance requests based on various parameters such as weighted load, client IP and server IP address-hash based format, or round-robin format.

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

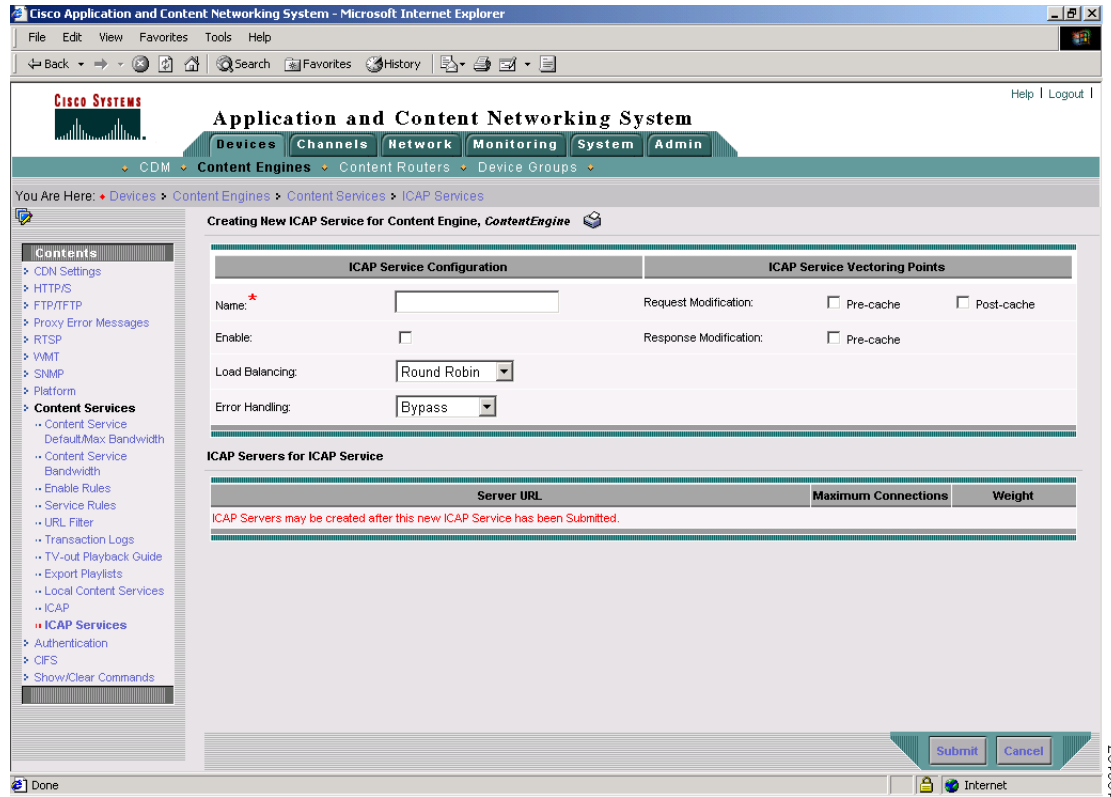
**Note**

When the Aggregate Settings option is applied, the ICAP services that have been previously configured for device groups to which the Content Engine belongs cannot be modified or deleted in the ICAP Services for Content Engine window. In other words, you can only view the ICAP services created for the device groups.

To configure an ICAP service for a Content Engine, follow these steps:

- Step 1** Choose **Devices > Content Engines**. The Content Engines window appears.
- Step 2** Click the **Edit** icon next to the desired Content Engine. The Modifying Content Engine window appears.
- Step 3** In the Contents pane, choose **Content Services > ICAP Services**. The ICAP Services for Content Engine window appears.
- Step 4** The **Yes** radio button for aggregate settings is selected by default, meaning that the ICAP services defined for the Content Engine and the device groups to which the Content Engine belongs are displayed. Alternatively, click the **No** radio button if you wish to display only the settings defined for the Content Engine.
- Step 5** Click the **Create New ICAP Service** icon in the taskbar. The Creating New ICAP Service for Content Engine window appears. (See [Figure 10-2](#).)

Figure 10-2 Creating New ICAP Service



- Step 6** Complete ICAP service configuration, as follows:
- a. In the Name field, enter a string identifying the ICAP service to be configured.
 - b. Check the **Enable** check box to enable ICAP service.
 - c. Choose the type of load-balancing mechanism for ICAP processing from the Load Balancing drop-down list. [Table 10-1](#) describes the load-balancing options.

Table 10-1 ICAP Load-Balancing Options

Load-Balancing Type	Description
Client IP Hash	Uses a hash-based algorithm based on the client IP address for load-balancing the ICAP servers in the cluster.
Round Robin	Uses the round-robin method, in which ICAP servers take turns processing HTTP requests.
Server IP Hash	Uses a hash-based algorithm based on the server IP address for load-balancing the ICAP servers in the cluster.
Weighted	Uses a farm of ICAP servers with different load capacities.

- Step 7** Choose the type of error handling mechanism for ICAP processing from the Error Handling drop-down list. If you wish to bypass this ICAP service, choose **Bypass**. Otherwise, choose **Return Error** if you want errors to be returned for client requests. These errors are also entered in the transaction log to show the status of the action performed by the ICAP services.

- Step 8** Under the ICAP Service Vectoring Points heading, configure the following options:
- Check the **Pre-cache** check box for request modification, to enable the vectoring point that is to be invoked when a Content Engine receives a request from a client.
 - Check the **Post-cache** check box for request modification, to enable the vectoring point that is to be invoked if the request is a cache miss and must be sent to the origin server for the content.
 - Check the **Pre-cache** check box for response modification, to enable the vectoring point that is to be invoked only when the response is from the origin server.
- Step 9** Click **Submit** to save your settings. The configured settings are saved to the database and the ICAP Services for Content Engine window appears, listing the configured ICAP service.
- Step 10** To display a subset of the entire list of ICAP services, click the **Filter Table** icon in the taskbar.
- Step 11** To revert to the display of all configured ICAP services, click the **View All ICAP Services** icon in the taskbar.
-

Configuring an ICAP Server for ICAP Service

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

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



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

To configure an ICAP server for a previously configured ICAP service on a Content Engine, follow these steps:

- Step 1** Choose **Devices > Content Engines**. The Content Engines window appears.
- Step 2** Click the **Edit** icon next to the desired Content Engine. The Modifying Content Engine window appears.
- Step 3** In the Contents pane, choose **Content Services > ICAP Services**. The ICAP Services for Content Engine window appears.
- Step 4** The **Yes** radio button for aggregate settings is selected by default, meaning that the ICAP services defined for the Content Engine and the device groups to which the Content Engine belongs are displayed. Alternatively, click the **No** radio button if you wish to display only the settings defined for the Content Engine.
- Step 5** Click the **Edit ICAP Service** icon next to the ICAP server for which you wish to configure an ICAP server. The Modifying ICAP Service for Content Engine window appears.
- Step 6** In the ICAP Servers for ICAP Service section, click the **Create new ICAP Server** icon in the taskbar. The Creating New ICAP Server for ICAP Service window appears.
- Step 7** Enter the host name or IP address of the ICAP server in the Server Host field.

- Step 8** In the Service Port field, enter the port number on which the ICAP server is to be configured to process HTTP requests. This step is optional. The default port number for ICAP server is 1344. Any valid port number can be set. If no port number is specified, the default port number is used for the ICAP server.
- Step 9** In the Server Service Name field, enter the path to the ICAP service configured on the Content Engine.
- Step 10** Enter the maximum number of simultaneous connections that can be made to this server in the Maximum Connections field. This step is optional. The valid range is 1 to 5000.
- Step 11** In the Weight field, enter the percentage of load that can be redirected to this ICAP server. This step is optional. The valid range is 1 to 100. This field must not be left blank if you chose the **Weighted** load-balancing method.
- Step 12** Click **Submit** to save your settings.

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

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

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

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

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

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

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

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

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

Alternatively, you can configure the Content Engine to bypass ICAP processing based on user agents, or any of the patterns available in the Rules Template, by using the **rule** command. In the following example, the Content Engine is configured to bypass ICAP processing on the intranet site `cisco.com` and on the trusted Internet site `datek.com`:

```
rule enable
rule action use-icap-service trend-reqmod pattern-list 1 protocol all
rule action use-icap-service trend-respmod pattern-list 1 protocol all
rule pattern-list 1 domain "!(.*cisco\.com|.*datek\.com)"
!
icap apply rules-template
icap service trend-reqmod
    enable
    vector-point reqmod-precache
```

```

server icap://172.19.227.150/REQ-Service
exit
icap service trend-respmod
enable
vector-point respmod-precache
server icap://172.19.227.150/interscan
exit

```

Configuring Service Rules

The Rules Template feature provides a flexible mechanism to specify configurable caching requests by allowing these requests to be *matched* against an arbitrary number of parameters, with an arbitrary number of *policies* applied against the matches. You can specify a set of rules, each clearly identified by an action and a pattern. Subsequently, for every incoming request, if a pattern for a rule matches the given request, the corresponding action for that rule is taken.

Requests can be matched against regular expressions symbolizing domain names, source IP addresses and network masks, destination IP addresses and network masks, destination port numbers, MIME types, or regular expressions symbolizing a URL.

You can configure service rules using the Content Distribution Manager GUI or CLI commands. This section provides instructions for using the Content Distribution Manager GUI. For more information on the types of policies that can be applied, actions and patterns, Rules Template processing, and using the CLI to configure service rules, refer to the *Cisco ACNS Software Command Reference, Release 5.1*.

To configure or modify service rule settings, you need to do the following:

- Enable rule settings for the Content Engine
- Configure service rules for the Content Engine

Enabling Rule Settings

Before you configure service rules, you need to enable rule settings for the Content Engine. To enable rule settings, follow these steps:

-
- Step 1** From the Content Distribution Manager GUI, choose **Devices > Content Engines**. The Content Engines window appears.
 - Step 2** Click the **Edit** icon next to the Content Engine that you want to configure. The Modifying Content Engine window appears.
 - Step 3** From the Contents pane, choose **Content Services > Enabling Rules**. The Enable Rule Settings window appears.
 - Step 4** Check the **Enable** check box to enable the use of rule settings.
 - Step 5** Click **Submit** to save the settings.
-

Configuring Service Rules

Configuring a service rule consists of the following tasks:

1. Configuring a pattern list
2. Adding a pattern to an existing pattern list
3. Associating an action with an existing pattern list

An action is a process that the Content Engine performs while processing the request, for example, blocking the request, redirecting the request, and so on. A pattern defines the limits of the request, for example, a pattern may specify that the IP address must fall within the subnet range 10.0.*.*.

To configure service rule parameters, follow these steps:

-
- Step 1** From the Content Distribution Manager GUI, choose **Devices > Device Groups**. If you have created device groups, the Device Group window appears.
 - Step 2** Click the **Edit** icon next to the name of the device group that you want to configure. The Contents pane appears on the left.
 - Step 3** From the Contents pane, choose **Content Services > Service Rules**.
 - Step 4** Click the **Create New Service Rule** icon. The Creating New Service Rule window appears.
 - Step 5** Configure a pattern list and add a pattern to it.
 - a. Choose **pattern-list** from the **Rule Type** drop-down list.
 - b. In the Rule Parameters field, configure the list number and the pattern type following the Rules Usage guidelines shown in the GUI. (See [Table 10-2](#) for a description of pattern types.)
For example, to create pattern list number 72 with the pattern type *domain* and the domain to be acted on by an action as yahoo.com, enter **72 domain yahoo.com** in the Rule Parameters field. (See [Figure 10-3](#).)

Figure 10-3 Creating a New Pattern List and Defining a Pattern

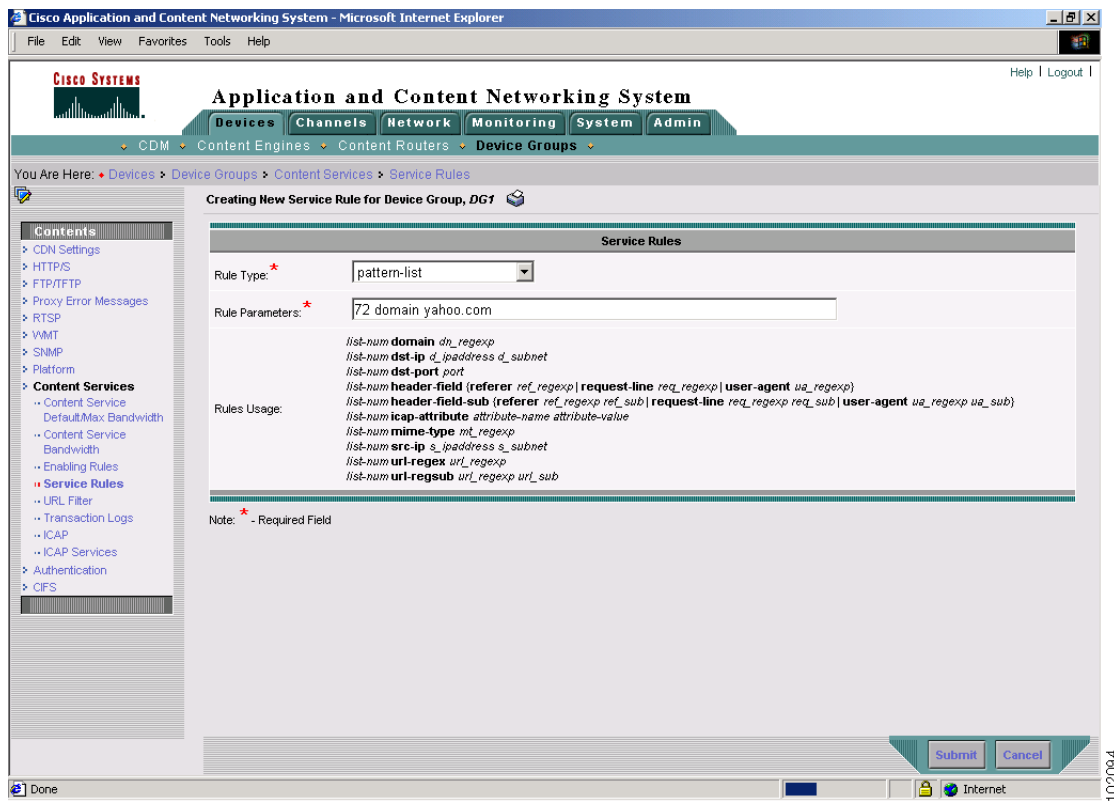


Table 10-2 Pattern Types

Pattern Types	Description
domain	<p>Matches the domain name in the URL or the Host header against a regular expression. For example, “.*ibm.*” matches any domain name that contains the “ibm” substring. “\.foo\.com\$” matches any domain name that ends with the “.foo.com” substring.</p> <p>In regular expression syntax, the dollar sign “\$” metacharacter directs that a match is made only when the pattern is found at the end of a line.</p>
dst-ip	<p>Matches the request’s destination IP address and netmask. Specify an IP address and a netmask. In proxy mode, the Content Engine does a DNS lookup to resolve the destination IP address of the HTTP request, making the response time longer, and possibly negating the benefit of setting a dst-ip rule. When an outgoing proxy is configured, cache miss requests are forwarded by the Content Engine to the outgoing proxy without examination of the destination server IP address, making the dst-ip rule unenforceable on the first Content Engine.</p>
dst-port	<p>Matches the request’s destination port number. Specify a port number.</p>

Table 10-2 Pattern Types (continued)

Pattern Types	Description
header-field	Request header field pattern. Request header field patterns referer , request-line , and user-agent are supported for actions block , reset , redirect , and rewrite . The referer pattern is matched against the Referer header in the request, the request-line pattern is matched against the first line of the request, and the user-agent pattern is matched against the User-Agent header in the request.
header-field-sub	Request header field subpattern.
icap-attribute	
mime-type	Matches the MIME type of the response. Specify a MIME type string, for example, “image/gif,” as defined in RFC 2046 (http://www.faqs.org/rfcs/rfc2046.html). The administrator can specify a substring, for example, “java” and have it apply to all MIME types with the “java” substring, such as “application/x-javascript.”
scr-ip	Matches the request’s source IP address and netmask. Specify an IP address and a netmask.
url-regex	Matches the URL against a regular expression. The match is case insensitive. Specify a regular expression whose syntax can be found at the following URL: http://yenta.www.media.mit.edu/projects/Yenta/Releases/Documentation/regex-0.12/ .
url-regexsub	For the rewrite and redirect actions, matches the URL against a regular expression to form a new URL per pattern substitution specification. The match is case insensitive. The valid substitution index range is from 1 to 9.

Step 6 Click **Submit**.

Step 7 Next, associate an action with an existing pattern list.

- a. From the Creating New Service Rule window (see [Step 1](#) through [Step 4](#)) choose an action type from the Rule Type drop-down list. (See [Table 10-3](#) for a description of rule actions.)
- b. In the Rule Parameter field, enter the list number of the pattern list that you want associated with this action.

For example, if you want to block access by any protocol to yahoo.com, then choose **block** from the Rule Type drop-down list and enter **pattern-list 72 protocol all** in the Rule Parameters field. (See [Figure 10-4](#).)

Figure 10-4 Associating an Action with an Existing Pattern List

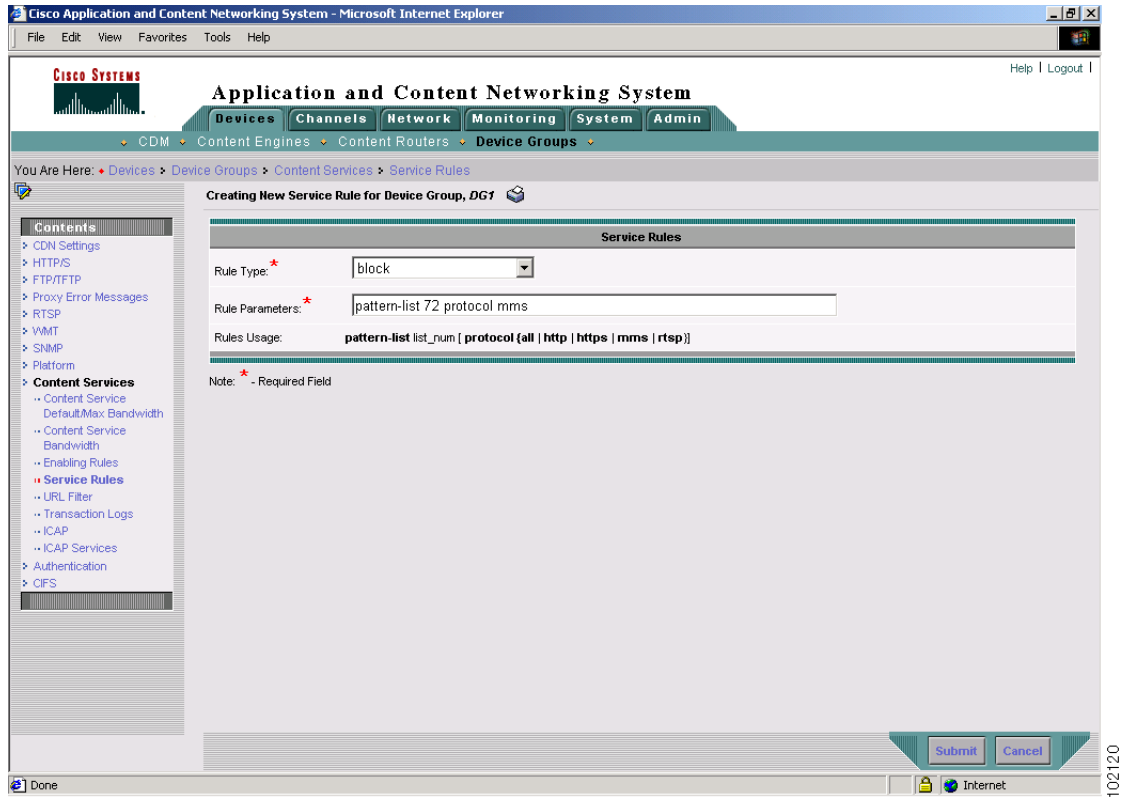


Table 10-3 Rule Actions

Action Types	Description
allow	Allows this request.
append-username-header	Appends the username in the request sent to the origin server.
block	Blocks this request.
cache-non-cacheable	<p>Overrides the HTTP response headers and caches the objects.</p> <p>This rule action caches objects only if they are not authenticated. That is, for authenticated objects, some origin servers do not send the Last-Modified and ETag entity headers. Revalidation of these objects therefore cannot be performed by the Content Engine. These authenticated objects are served only from the origin server.</p> <p>If the server <i>does</i> send the Last-Modified and ETag headers, then these objects can be cached.</p>

Table 10-3 Rule Actions (continued)

Action Types	Description
cache-only	<p>Caches objects depending on the HTTP response headers. Caches this object only if it is a match and is allowed to be cached by HTTP.</p> <p>If one or more rules specify this action, an object is cached if and only if it matches at least one of the selective-cache rules and passes every other caching restriction such as the object-size check and the no-cache-on-authenticated-object check.</p> <p>If the object does not match any of the selective-cache rules, the object is not cached.</p>
dscp	<p>Configures the IP ToS or DSCP code point field.</p> <p>cache-miss—Sets the IP ToS/DSCP code point bits for the client-side connection to the configured value for cache miss responses to the client.</p> <p>cache-hit—Sets the IP ToS or DSCP code point bits for the client-side connection to the configured value for cache hit responses to the client.</p> <p>Setting the Type of Service (ToS) or differentiated services code point (DSCP) is called packet marking, allowing you to partition network data into multiple priority levels or types of service. With the ACNS 5.x releases, 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.</p> <p>You can set specific ToS or DSCP values for the following:</p> <ul style="list-style-type: none"> • Requests from the Content Engine to the server • Responses to the client on a cache hit • Responses to the client on a cache miss <p>The ToS or DSCP may be set based on any of the policies matching the src-ip <i>s_ipaddress s_subnet</i>, dst-ip <i>d_ipaddress d_subnet</i>, dst-port <i>port</i>, domain <i>LINE</i>, url-regex <i>LINE</i>, or mime-type <i>LINE</i> options. In addition, you can now configure global ToS or DSCP settings with the ip dscp command.</p> <p>The Rules Template configuration takes precedence over the ip dscp command, and the url-filter command takes precedence over the rule command to the extent that even the rule no-block command is executed only if the url-filter command has not blocked the request.</p>
freshness-factor	<p>Determines the Time To Live if the request URL matches a specified regular expression. The refresh configuration takes priority over freshness-factor configurations.</p>
insert-no-cache	<p>Inserts a no-cache header in the response.</p>
no-auth	<p>Does not authenticate.</p>
no-cache	<p>Does not cache this object. If both the no-cache and selective-cache actions are matched, no-cache takes precedence.</p>

Table 10-3 Rule Actions (continued)

Action Types	Description
no-proxy	For a cache miss, does not use the configured upstream proxy but rather contacts the server directly.
redirect	Redirects the original request to a specified URL. Redirect is relevant to the RADIUS server only if the RADIUS server has been configured for redirect.
redirect-url-for-cdn	Redirects the original request to a specified URL for the ACNS network.
refresh	For a cache hit, forces an object freshness check with the server.
reset	Issues a TCP RST. This reset request is useful when resetting Code Red or Nimda virus requests.
rewrite	Rewrites the original request as a specified URL
use-dns-server	Caches this object only if it is a match and is allowed to be cached by HTTP. If one or more rules specify this action, an object is cached if and only if it matches at least one of the selective-cache rules and passes every other caching restriction such as the object-size check and the no-cache-on-authenticated-object check. If the object does not match any of the selective-cache rules, the object is not cached.
use-icap-service	Applies ICAP processing and uses a specific ICAP service only to those requests that match this rules template action. An ICAP service is a collection of attributes that defines the type of modification to be performed on HTTP requests and responses. If this action is configured, you can allow requests and responses to be processed by ICAP servers for content adaptation
use-proxy	For a cache miss, uses a specific upstream proxy. Specify the upstream proxy IP address (or domain name) and port number. If both no-proxy and use-proxy are matched, no-proxy takes precedence.
use-server	Sends server-style HTTP requests from the Content Engine to the specified IP address and port on a cache miss.
use-xforward-clt-ip	Uses client IP address in the forwarded header for filtering.

Step 8 Click **Submit** to save the settings.

Configuring the Rules Template Using CLI Commands

These sections describe how to configure pattern lists and actions for the Rules Template using CLI commands.

- [Configuring a Pattern List](#)
- [Adding a Pattern to an Existing Pattern List](#)
- [Associating an Action with an Existing Pattern List](#)

Configuring a Pattern List

To create a new pattern list, follow these steps:

	Command	Purpose
Step 1	ContentEngine(config)# rule enable	Enables the Rules Template.
Step 2	ContentEngine(config)# rule pattern-list <1-512>	Creates a pattern list.
Step 3	ContentEngine# show rule pattern-list <1-512> pattern-type <pattern>	Displays the Rules Template configuration.

In the following example, the **rule pattern-list** command is configured to create a pattern list to block all domains that contain .foo.com in the URL request using the domain \.foo.com pattern.

```
ContentEngine(config)# rule pattern-list 10 domain foo.com
ContentEngine# show rule pattern-list 10 domain
Rules Template Configuration
-----
Rule Processing Enabled

Pattern-Lists :

rule pattern-list 10 domain foo.com
ContentEngine#
```

Adding a Pattern to an Existing Pattern List

To add a new pattern to an already existing pattern list, follow these steps:

	Command	Purpose
Step 1	ContentEngine(config)# rule pattern-list <1-512> pattern-type <pattern>	Adds a pattern to a pattern list.
Step 2	ContentEngine# show rule pattern-list <1-512> pattern-type <pattern>	Displays the Rules Template configuration.

In the following example, the **rule pattern-list** command is configured to add a pattern to an existing pattern list to perform an action yet to be defined on the destination IP address 172.16.25.25 using the **dst-ip** pattern.

```
ContentEngine(config)# rule pattern-list 10 dst-ip 172.16.25.25 255.255.255.0
ContentEngine# show rule pattern-list 10 all
Rules Template Configuration
-----
```

```

Rule Processing Enabled

Pattern-Lists :

rule pattern-list 11 dst-ip 172.16.25.25 255.255.255.0
rule pattern-list 11 domain foo.com
ContentEngine#

```

Associating an Action with an Existing Pattern List

To associate an action with an existing pattern list, follow these steps:

	Command	Purpose
Step 1	ContentEngine(config)# rule action <action_type> pattern-list <1-512> protocol <protocol_type> all	Associates an action with an existing pattern list.
Step 2	ContentEngine# show rule action <action_type> protocol <protocol_type> all	Displays the Rules Template configuration.

In the following example, the **rule action block** command is configured and associated with an existing pattern list.

```

ContentEngine(config)# rule action block pattern-list 10 protocol all
ContentEngine# show rule action block
Rules Template Configuration
-----
Rule Processing Enabled

Actions :

rule action block pattern-list 10 protocol all
ContentEngine#

```

Verifying an Action Performed on a Pattern List

To verify the response sent by the Content Engine to confirm that a certain action is performed on a pattern list, follow these steps:

	Command	Purpose
Step 1	ContentEngine(config)# rule action <action_type> pattern-list <1-512> protocol <protocol_type> all	Associates an action to an existing pattern list.
Step 2	ContentEngine# show rule action <action_type> protocol <protocol_type> all	Displays the Rules Template configuration after a new action has been added.
Step 3	ContentEngine# show statistics rule action <action_type>	Displays the local Rules Template configuration statistics after a request is issued on which an action should be performed.

In the following example, the **rule action block** command is configured and associated with an existing pattern list, which lists as its pattern the domain yahoo.com.

```

ContentEngine(config)# rule pattern-list 30 domain yahoo.com
ContentEngine(config)# rule action block pattern-list 30 protocol all
ContentEngine# show statistics rule action block

```

```
Rules Template Statistics
-----
Rule hit count = 3   Rule: rule action block pattern-list 30 protocol all
ContentEngine#
```

In this example, the request to yahoo.com was denied three times.

Configuring URL Filtering

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.

The URL filtering features presented in this section allow the Content Engine to control client access to websites in any of the following ways:

- Deny access to URLs specified in a list (HTTP, MMS, and RSTP traffic).
- Permit access only to URLs specified in a list (HTTP, MMS, and RSTP traffic).
- Direct traffic to an N2H2 server for filtering (HTTP traffic only).
- Direct traffic to a Websense enterprise server for filtering (HTTP traffic only).

For information about configuring the Websense software, go to the following website:

<http://www.websense.com>

- Filter traffic with Secure Computing Corporation SmartFilter Software, Release 3.2 (HTTP traffic only).

For information about configuring the SmartFilter software, go to the following website:

<http://www.securecomputing.com>



Note

Although only one form of URL filtering scheme per protocol can be active, many URL filtering schemes can be supported at one time. In other words, if an N2H2 filter is applied to HTTP URLs, no other URL filtering scheme, such as Websense, or SmartFilter, can be applied to this protocol. However, the use of good and bad lists can be applied to the streaming media protocol. The scheme enabled for a particular protocol is independent of that of other protocols.

Custom Blocking Messages

The Content Engine can be configured to return a customized blocking message to the client. The custom message must be an administrator-created HTML file named *block.html*. Make sure to copy all embedded graphics associated with the custom message HTML window to the same directory that contains the *block.html* file. To enable the customized blocking message, use the **url-filter http custom-message** command and specify the directory name.

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

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

**Note**

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

In the example shown, a `block.html` file displays the following custom message:

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

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

In the `block.html` file shown, 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 example below.

**Note**

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

```
<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>
```

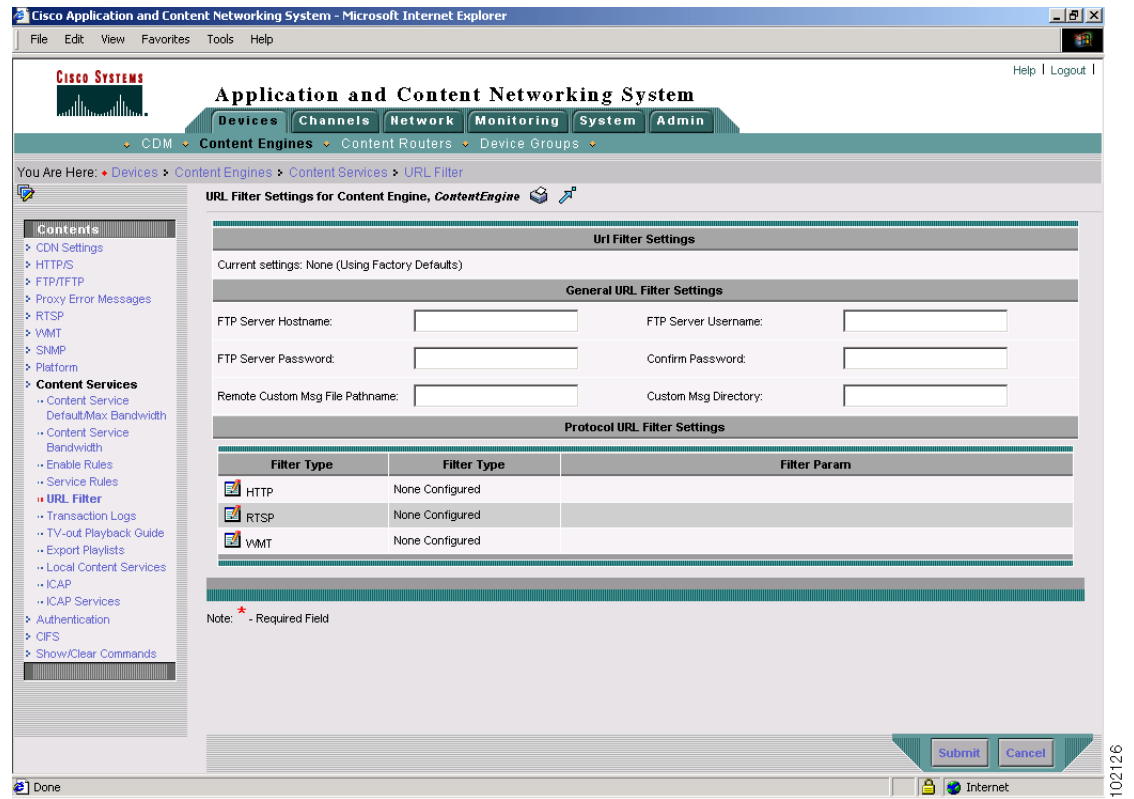
If the `block.html` file is updated, it will automatically display its new message without your having to reenter the `url-filter http custom-message` command.

Configure URL Filtering Using the Content Distribution Manager GUI

To configure URL filter settings for the Content Engine, follow these steps:

- Step 1** From the Content Distribution Manager GUI, choose **Devices > Content Engines**. The Content Engines window appears.
- Step 2** Click the **Edit** icon next to the Content Engine that you want to configure. The Contents pane appears on the left.
- Step 3** Click **Content Services > URL Filter**. The URL Filter Settings window appears. (See [Figure 10-5](#).)

Figure 10-5 URL Filter Settings Window



- Step 4** Enter information for general URL filter settings. (See [Table 10-4](#) for descriptions of the filter-setting parameters.)
- Step 5** Under the Protocol URL Filter Settings heading, click the **Edit** icon next to the name of the filter type and enter information for HTTP URL filter settings, RTSP URL filter settings, or WMT filter settings in the appropriate fields. (See [Table 10-4](#).)
- Step 6** Click **Submit** to save the settings.

Table 10-4 URL Filter Settings Window

Key GUI Parameter	Description	CLI Command
General URL Filter Settings		
FTP Server Hostname	DNS name or IP address of the FTP server from which the URL filter files are downloaded.	—
FTP Server Username	Name needed to access the FTP server from which the URL filter files are downloaded.	—
FTP Server Password	Password of the FTP server from which the URL filter files are downloaded.	—
Confirm Password	Confirms the FTP server password.	—

Table 10-4 URL Filter Settings Window (continued)

Key GUI Parameter	Description	CLI Command
Remote Custom Msg File Pathname	Path name of the remote file that contains the custom message directory.	—
Custom Msg Directory	Creates a customized URL blocking message to display to the client. This custom message must be an administrator-created HTML file named <i>block.html</i> .	url-filter http custom-message dirname
Protocol URL Filter Settings for HTTP, RTSP, and WMT		
Enable Bad Site Filtering	Enables the use of local list filtering for bad sites.	url-filter http bad-sites-deny enable url-filter rtsp bad-sites-deny enable url-filter wmt bad-sites-deny enable
Remote Bad Site File Pathname	Path name of remote bad site file.	—
Bad Site Filename	File containing URLs to which access is denied.	url-filter http bad-sites-deny file filename url-filter rtsp bad-sites-deny file filename url-filter wmt bad-sites-deny file filename
Enable Good Site Allow	Enables URL filtering of the local good sites list over HTTP.	url-filter http good-sites-allow enable url-filter rtsp good-sites-allow enable url-filter wmt good-sites-allow enable
Remote Good Site File Pathname	Path name of remote good site file.	—

Table 10-4 URL Filter Settings Window (continued)

Key GUI Parameter	Description	CLI Command
Good Site Filename	File containing URLs to which access is permitted.	url-filter http good-sites-allow file filename url-filter rtsp good-sites-allow file filename url-filter wmt good-sites-allow file filename
FTP Server Hostname	DNS name or IP address of the FTP server from which the URL filter files are downloaded.	—
FTP Server Username	Name needed to access the FTP server from which the URL filter files are downloaded.	—
FTP Server password	Password of the FTP server from which the URL filter files are downloaded.	—
Confirm password	Confirms the FTP server password.	—
Enable SmartFilter Filtering	Enables the use of SmartFilter software.	url-filter http smartfilter enable
Enable N2H2 Filtering	Enables the use of an N2H2 server for URL filtering.	url-filter http N2H2 enable
N2H2 Server Hostname	Host name or IP address of the N2H2 server.	url-filter http N2H2 server
N2H2 Port	Port number on which the N2H2 server is accepting requests.	url-filter http N2H2 server IPaddress or hostname port
Enable N2H2 Allow Mode	Allows the request to be served if there is no response from the N2H2 server. Required check box.	url-filter http N2H2 allowmode enable
N2H2 Request Timeout	Number of seconds that the Content Engine should wait for a response from the N2H2 server.	url-filter http N2H2 server IPaddress or hostname port 1-65535 timeout
Enable Websense Filtering	Enables the use of a Websense server for URL filtering.	url-filter http websense enable
Use Embedded WebSense Server	Configures the Websense server on the Content Engine. Ensures that the URL filtering software uses the local Websense server and not a remote host as the Websense server.	—
WebSense Server Hostname	Host name or IP address of the Websense server	url-filter http websense server hostname

Table 10-4 URL Filter Settings Window (continued)

Key GUI Parameter	Description	CLI Command
WebSense Port	Port number on which the Websense server is accepting requests.	url-filter http websense server hostname or IP address port 1-65535
Enable WebSense Allow Mode	Allows the request to be served if there is no response from the Websense server.	url-filter http websense allowmode enable
WebSense Request Timeout	Number of seconds the Content Engine should wait for a response from the Websense server.	url-filter http websense server IP address or hostname port 1-65535 timeout
Websense Request Connections	Number of persistent connections to the Websense server. The range is 1–250 connections per CPU. The default is 40 connections. Do not change the default unless you know for certain that a different value is required.	url-filter http websense server IP address or hostname port 1-65535 timeout seconds connections 1-250

URL Filtering with URL Lists

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

The use of URL lists applies to requests in HTTP, HTTPS, and FTP format as well as streaming media protocols such as MMS and RTSP.



Note

The local list file per protocol should not contain URLs that belong to other protocols. For instance, the HTTP local list file should only contain 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 is too large, it can affect device performance, because the file is loaded into memory when local list file filtering is enabled. If the size of the file is larger than 5 megabytes, a warning is issued by the device to notify you of its impact on performance.

To deny 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 delimited with carriage returns.

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



Note

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 file** 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** command to actively deny the URLs.

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

To 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 delimited with carriage returns.

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



Note 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** 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** command to actively permit only the good URLs.

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



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



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

Use the **no** form of the command to disable blocking, Websense, or N2H2 permission requests (for example, **no url-filter bad-sites-deny**).

URL Filtering with the N2H2 Server

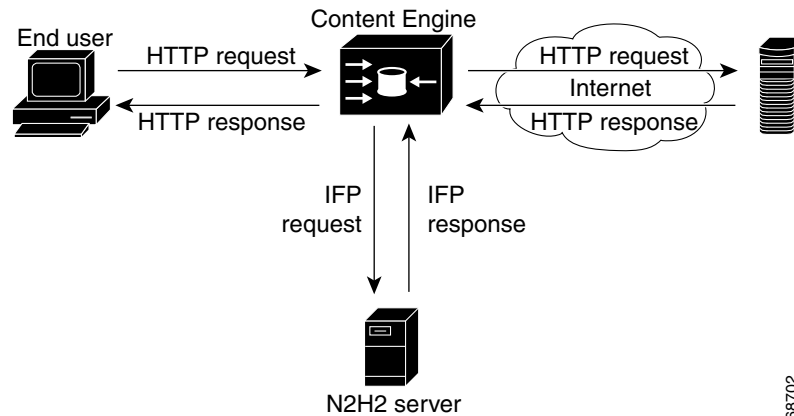


Note URL filtering with the N2H2 server, Websense server, and SmartFilter software only applies to HTTP, FTP, or HTTPS protocols.

N2H2 is a globally deployed URL-filtering software that can filter HTTP, FTP, or HTTPS requests based on destination host name, destination IP address, and username 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.

The Content Engine can perform URL filtering based on the N2H2 server. (See [Figure 10-6](#).) The Content Engine and the N2H2 server use the Internet Filtering Protocol (IFP) Version 1 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 block page or proceeds with normal HTTP processing by sending the URL request to an origin server.

Figure 10-6 N2H2 Filtering



URL filtering using an N2H2 server is applied to HTTP, FTP, or HTTPS traffic before the service rule mechanism is applied, regardless of whether the requested object is in the cache or not. Filtering is applied to these traffic types:

- Proxy-style or transparent-style HTTP or HTTPS requests
- Proxy-style and transparent redirect proxy-style FTP over HTTP requests

N2H2 Features Supported

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 N2H2 Features Supported

N2H2 Feature Name	Description
Global filtering	Applies filtering to all HTTP, FTP, or HTTPS 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 the IFP responses.

N2H2 CLI Commands

The **url-filter http N2H2 server** *IP address* [**port** 1-65535] [**timeout** 1-120] command configures the Content Engine to query the N2H2 server. The optional **port** field specifies the port on the N2H2 server to which the Content Engine should send IFP requests. The default port number is 4005. The optional **timeout** field (in seconds) specifies how long the Content Engine should wait for an IFP response from the N2H2 server. The default timeout is 5 seconds.

This 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.

The **url-filter http N2H2 enable** command enables the N2H2 server as the current URL filtering scheme. The command will not succeed if the server IP address is not configured, or if another URL filter is already enabled with N2H2 or other filtering schemes.

The **url-filter http N2H2 allowmode enable** command allows HTTP or HTTPS requests to pass when the N2H2 server is enabled but the Content Engine has problems communicating with the N2H2 server. With **allowmode** enabled, if the Content Engine fails to receive responses from the N2H2 server successfully, the Content Engine still allows all traffic through (it proceeds with normal traffic processing). With **allowmode** disabled, on the other hand, the Content Engine blocks all traffic through the Content Engine. By default, **allowmode** is enabled.

The **allowmode** option can be configured with or without N2H2 enabled and is independent of the N2H2 server configuration. The Content Engine adopts the new configuration for **allowmode** if N2H2 is already running.

N2H2 Status and Statistics Commands

The **show url-filter http** command shows the URL filtering scheme enabled on the Content Engine and the configurations for each URL filtering scheme, such as the configuration data for N2H2.

In this example, the **show url-filter http** command is used to display the status of all HTTP URL filtering schemes presently configured on the 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 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#
```

The **show statistics url-filter http n2h2** command shows the request-reply statistics of 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.

```
ContentEngine# show stat url-filter http n2h2
N2H2 URL Filtering Statistics:
    Lookup requests transmitted = 144
    Lookup response received = 144
    Requests timed out = 0
    Number of retransmits = 0

    Requests BLOCKed by N2H2 = 52
    Requests OKed by N2H2 = 92

Allow Mode Statistics:
    No available connection = 0
    Error sending lookup requests = 0
    Error receiving lookup responses = 0
    Server error in responses = 0

Server Error in Responses:
    Error in Filter Server = 0
    Error in IFP server = 0
    Seq number mismatch = 0
    Multiple responses rcvd = 0

TCP error statistics:
    Bad network endpoint = 0
    Network unreachable = 0
    Underlying connection broken = 0
    Timeout specified is reached = 0
    Address already in use = 0
    Client connection broken = 0
    Client connection timeout = 0
    Server connection broken = 0
    Server connection timeout = 0
    Register read cancelled = 0
    Other errors = 0

Queue statistics:
    Number of xacts in Queue = 0

Overhead statistics:
    Avg total process time = 17
    Avg response time = 17
    Socket update count = 0

ContentEngine#
```

The statistics shown can be cleared using the **clear statistics url-filter N2H2**, **clear statistics urlfilter**, and **clear statistics all** commands.

The **clear statistics url-filter N2H2** command resets the statistics counters for the N2H2 server. All the statistics counters are reset to 0.

N2H2 Configuration and Restrictions

Only one URL filtering scheme can be active per protocol. In order to enable N2H2 URL filtering, you should first make sure that no other URL filtering scheme is configured. You can then configure the server information for N2H2 using the **url-filter N2H2 server IP address [port 1-65535] [timeout 1-120]** command and enabling the N2H2 server.

The server IP address and port number configured in the Content Engine must match the IP address of the N2H2 server and the port that 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, FTP, or HTTPS requests and either block or allow all HTTP traffic based on the **allowmode** option configuration.

URL Filtering with the Integrated Websense Server

Cisco ACNS 5.1 software supports Websense server Version 5.0.1 on all Cisco Content Engine platforms. The integrated Websense server runs internally to the Content Engine (as opposed to running on a separate system and communicating with the Content Engine over the network) and uses approximately 60 MB to 140 MB of RAM in the Content Engine. We recommend that you run the integrated Websense server on Content Engines with at least 512 MB of RAM for best results.

When the Websense server is enabled and the Websense URL database is downloaded the first time, CPU usage can be very high. Therefore, we recommend that you enable Websense server during off-peak times or times of low network traffic. Otherwise, other processes running on the Content Engine might be affected. If the Websense server stalls, it restarts automatically.

Websense provides an image of the Websense server that resides in the `/local/local1/WebsenseEnterprise` directory. All the executables as well as the configuration and logging files are stored in the above mentioned directory. This package requires about 150 MB of disk space in the `/local/local1/WebsenseEnterprise/EIM` directory. An additional 140 MB of disk space is required at the time of downloading the Websense URL database, increasing the total disk space requirement to 290 MB. To ensure that you have enough disk space to properly download the Websense software, we recommend that you increase the amount of sysfs disk space to be greater than the default sysfs on your Content Engine.

Configuring Ports for the Websense Server

The integrated Websense process requires that four ports be opened for connections either from processes internal to the Content Engine or from external processes such as the PIX firewall. These four ports and default port numbers are as follows:

- 15868—Websense server port
This is the TCP port that receives requests for content filtering according to the Websense protocol.
- 15871—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.
- 15870—Configuration server port
This port is required by the Websense GUI to configure the Websense server.

- 15869—Diagnostics server port

The Websense server has an exhaustive set of diagnostics that the users can run remotely to diagnose problems in the Websense process. This port is the one that these diagnostics utilities connect to.

You can configure these ports by modifying the websense.ini file which resides in the /local/local1/WebsenseEnterprise directory. 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 websense.ini file using FTP from the /local/local1/WebsenseEnterprise directory on the Content Engine, modifying the file, deleting the websense.ini file on the Content Engine, and then sending back the modified file to the Content Engine using FTP.

**Note**

Websense server needs to be disabled and enabled to pick up the newly configured ports.

Enabling the Integrated Websense Server Using the Content Distribution Manager GUI

To enable the integrated Websense server using the Content Distribution Manager GUI, follow these steps:

-
- Step 1** From the Content Distribution Manager GUI, choose **Devices > Content Engines**.
 - Step 2** Click the **Edit** icon next to the Content Engine that you want to view. The Modifying Content Engine window appears.
 - Step 3** In the Contents pane, choose **Content Services > URL Filter**. The URL Filter Settings for Content Engine window appears. (See [Figure 10-5](#).)
 - Step 4** Click the **Edit** icon next to the HTTP filter type to configure and enable the Websense server to run on the Content Engine. The URL Filter Settings for Content Engine window refreshes itself, with the HTTP URL Filter Settings options activated. (See [Figure 10-7](#).)

Figure 10-7 URL Filter Settings—HTTP URL Filter Settings

The screenshot shows the Cisco Application and Content Networking System web interface. The main configuration area is titled "URL Filter Settings for Content Engine, ContentEngine". The settings are as follows:

FTP Server Username:	<input type="text"/>	FTP Server password:	<input type="password"/>	Confirm password:	<input type="password"/>
Enable Bad Site Filtering:	<input type="checkbox"/>	Remote Bad Site File Pathname:	<input type="text"/>	Bad Site Filename: *	<input type="text"/>
Enable Good Site Allow:	<input type="checkbox"/>	Remote Good Site File Pathname:	<input type="text"/>	Good Site Filename: *	<input type="text"/>
Enable SmartFilter Filtering:	<input type="checkbox"/>	Enable N2H2 Filtering:	<input type="checkbox"/>	N2H2 Server HostName: *	<input type="text"/>
				N2H2 Port: *	<input type="text" value="4005"/>
		Enable N2H2 Allow Mode: *	<input checked="" type="checkbox"/>	N2H2 Request Timeout: *	<input type="text" value="5"/>
Enable WebSense Filtering:	<input type="checkbox"/>	Use Embedded WebSense Server:	<input type="checkbox"/> (CE-7305 and CE7325 only)	WebSense Server HostName: *	<input type="text"/>
				WebSense Port: *	<input type="text" value="15868"/>
		Enable WebSense Allow Mode: *	<input checked="" type="checkbox"/>	WebSense Request Timeout: *	<input type="text" value="20"/>

At the bottom right of the configuration area, there are "Submit" and "Cancel" buttons. The status bar at the bottom shows "Done" and "Internet".

- Step 5** Check the **Enable WebSense Filtering** check box to enable URL filtering using the Websense server.
- Step 6** Check the **Use Embedded WebSense Server** check box to configure the Websense server on the Content Engine. This ensures that the URL filtering software uses the local Websense server and not a remote host as the Websense server. The Websense Server Hostname field becomes inactive.
- Step 7** In the Websense Port and Websense Request Timeout fields, leave the default settings as they appear unless different settings are required.
- Step 8** Check the **Enable WebSense Allow Mode** check box to enable HTTP access to a website if the Websense server does not respond.
- Step 9** In the WebSense Request Connections field, leave the default setting at 40 connections per CPU unless you know for a certain that a different value is required.
- Step 10** Click **Submit** to confirm your settings.

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

<http://www.websense.com/downloads>

To access the set of documents on Websense product setup and implementation, you can access the following URL:

<http://www.websense.com/support/documentation/index.cfm>

Integrated Websense Server CLI Commands

CLI commands for the integrated Websense server are as follows:

- **url-filter http websense server** {*local* | *hostname*}

If the *local* variable is specified during the configuration, then the Content Engine sends URL filtering requests to the Websense server running on the Content Engine. On the other hand, if the *hostname* variable is chosen, then the Websense server running on the Content Engine is not used.

- **websense-server enable**

This global configuration command enables the local Content Engine to act as the Websense server. When this command is used, a back-end script starts the integrated Websense server process through the node manager.

If the default ports are changed, the integrated Websense server must be disabled and re-enabled before the changes can be implemented.

- **show url-filter http**

This command shows the IP address of the local host in the Websense sever IP field when the local host is configured as the Websense server for Websense URL filtering.

- **show websense-server**

This command shows the configuration for the Websense server configured on the Content Engine. The output of the command includes the configured port numbers for the Websense server port, blocked message server port, configuration server port, and diagnostics server port; the Websense server version number; and the maximum number of connections.

- **write memory**

This EXEC command saves modified Websense configuration files (*websense.init* and *ws.cfg*) across disk reconfiguration and ACNS software release upgrades.

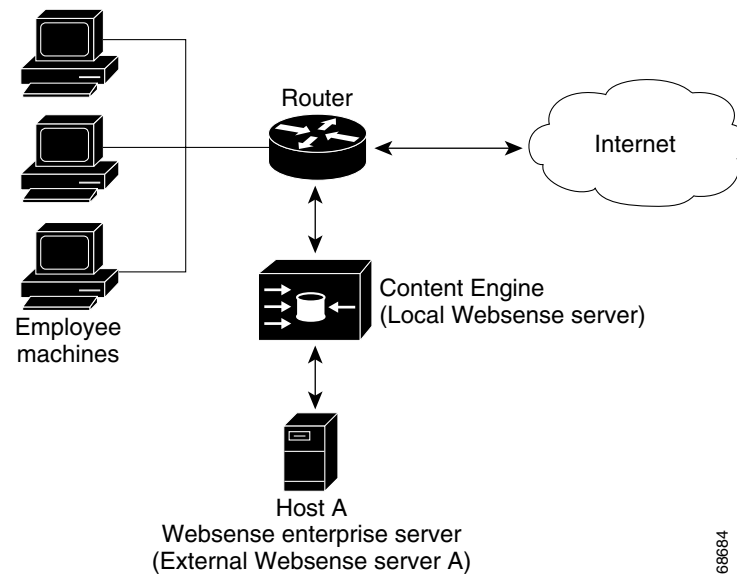
You must execute this command in order to have the most recent configuration modifications, including *websense.ini* file modifications and the Websense URL filtering configuration changes. The **write memory** command enables the changes made from the external Websense Manager GUI to be saved across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

If the **write memory** command is not used before reboot but after a disk reconfiguration or an ACNS software upgrade that erases disk content, the Websense configurations that were saved when the **write memory** command was last used are retained. However, if the **write memory** command was never used before, then default configurations are applied when the content on */local/local1/WebsenseEnterprise* directory is erased.

URL Filtering with the Websense Enterprise Server

The Content Engine can use a Websense enterprise server as a filtering engine and enforce the filtering policy configured on the Websense server. (See [Figure 10-8](#).) Refer to the Websense documentation for further information on Websense filtering policies.

Figure 10-8 URL Filtering with a Websense Server



Websense CLI Commands

CLI commands for using the Websense Enterprise server are as follows:

- **url-filter http websense enable**

To enable Websense URL filtering on the Content Engine, specify the Websense server IP address or host name using the **url-filter http websense enable** command. The **timeout** option sets the maximum amount of time that the Content Engine will wait for a Websense response. The timeout default is 20 seconds. The **port** option specifies the port number on which the server will accept requests from the Content Engine (the default port is 15868). Use the **no url-filter http websense enable** command to disable Websense URL filtering.



Note To use Websense URL filtering with a cluster of Content Engines, make sure to enable it with the **url-filter http websense enable** command, and configure the **url-filter http websense server** command on each Content Engine in the cluster to ensure that all traffic is filtered.

- **url-filter http websense allowmode enable**

The **url-filter http websense allowmode enable** command permits the Content Engine to fulfill the client request after a Websense server timeout. The Websense server returns its own blocking message when a request is denied. With **allowmode** disabled, on the other hand, the Content Engine blocks all traffic through the Content Engine. By default, **allowmode** is enabled.

Websense Status and Statistics Commands

The **show url-filter http** command shows the URL filtering scheme enabled on the Content Engine for HTTP traffic and the configurations for each URL filtering scheme, such as the configuration data for Websense.

In this example, the **show url-filter http** command is used to display the status of all HTTP URL filtering schemes presently configured on the 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 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 DISABLED
ContentEngine#
```

The **show statistics url-filter http websense** command shows the request-reply statistics of 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.

```
ContentEngine# show statistics url-filter http websense
Websense URL Filtering Statistics:
Transmission statistics:
    Lookup requests transmitted = 1
    Lookup requests timed-out = 1
    Lookup responses received = 1
    Lookup responses received with error = 0
    Multiple response received = 0
    Sequence number mismatch = 0

TCP errors:
    Connection reset = 0
    Connection timeout = 3
    Other errors = 0

Filter results:
    Requests BLOCKed by Websense = 1
    Requests OKed by Websense = 0
    Sent to Allowmode ok = 1
    Sent to Allowmode block = 0
    desc_filtered_and_passed = 0
    desc_category_blocked = 1
    desc_category_not_blocked = 0
    desc_category_blocked_custom_deny = 0
    desc_category_not_blocked_custom_permit = 0
```

```
 Websense log statistics:
      Logs sent successfully = 1
      Connection error = 0
      Error during log processing = 0
      Log not complete = 1
      Log not sent because Websense disabled = 0
      No available connection = 0

 Congestion statistics:
      Pending requests = 0
      Pending log requests = 0

 ContentEngine#
```

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

Websense Configuration and Restrictions

Only one URL filtering scheme can be active per protocol. In order to enable Websense URL filtering, you should first make sure that no other URL filtering scheme is enabled on the same protocol. You can then configure the information for the Websense server using the **url-filter http websense server IP address [port 1-65535] [timeout 1-120]** command and enabling the Websense server with the **url-filter http websense enable** command.

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

URL Filtering with SmartFilter Software

SmartFilter software for the Content Engine provides employee Internet management (EIM) functionality with proxy servers, firewalls, and caching appliances. The integrated Content Engine and SmartFilter product preserves all functionality available in a regular Content Engine. The SmartFilter filtering capability is available as an add-on service on the Content Engine, and the service may be enabled or disabled as desired through the Content Engine CLI or GUI.

The integrated Content Engine and SmartFilter product 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.

The integrated Content Engine and SmartFilter product provides two end user management tools called the SmartFilter Administration Console and the SmartFilter Administration Server. This GUI components download configurations into the Content Engine for use by the SmartFilter process.

To use SmartFilter URL filtering with a cluster of Content Engines, make sure to enter the **url-filter http smartfilter enable** command on each Content Engine in the cluster to ensure that all traffic is filtered.

Refer to the *SmartFilter for Cisco Content Engine User's Guide, Release 3.1* for more information on how to configure the SmartFilter software, and visit the SecureComputing website (<http://www.securecomputing.com>) for the most current SmartFilter documentation.

