CHAPTER 16

Configuring Request Processing 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

This section describes how to configure the Internet Content Adaptation Protocol (ICAP) on centrally managed Content Engines. The ACNS 5.2.1 software and later releases support ICAP for HTTP requests. The ACNS 5.4.1 software and later releases support ICAP for FTP-over-HTTP requests. Support for native FTP requests is not supported.

Overview of ICAP

ICAP is an open standards protocol that can be used for content adaptation, typically at the network edge. Content adaptation includes virus scanning, content translation, content filtering, content insertion, and other ways of improving the value of content to end users. ICAP specifies how a Content Engine, acting as an HTTP proxy server, can communicate with an external device that is acting as an ICAP server, which filters and adapts the requested content.

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

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

About ICAP Services

An ICAP service is a collection of attributes that define the service and one or more ICAP servers that provide the ICAP services. 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 on all HTTP requests processed by the Content Engine or apply ICAP processing only to requests that match the Rules Template.

Tip To set the type of load balancing to use among a cluster of ICAP servers, use the **icap service load balancing** global configuration command.

About ICAP Services and Vectoring Points

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

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

- Caches the response using the original URL
- Caches the response using an alternative URL

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

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

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

If an ICAP vendor supports the same service name for more than one vectoring point, you can configure a single service and add the supported vectoring points, as in the following example:

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

About ICAP Performance

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

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

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

ICAP Vendors Supported

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

- TrendMicro for reqmod and respmod
- Symantec for respmod

Maximum File Size Supported

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

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

For ACNS 5.5.9 software and later, files that exceed 2 GB are supported only if the request has content-length and **icap respmond** is not enabled. This applies only to the HTTP protocol. HTTPS and FTP-over-HTTP protocols are not supported.

Enhancements to ICAP Support in ACNS 5.5 Software

The ACNS 5.5.1 software supports the following enhancements for ICAP support:

- ICAP request modification support for HTTPS requests

  The ACNS 5.5.1 software supports an ICAP request modification (reqmod) for the HTTPS proxy-style requests. Support for ICAP reqmod processing allows proxy-style requests that use the HTTPS protocol to be modified as they are sent from the Content Engine to the ICAP server on their way to the origin server.

- Compatibility with the data trickling feature of the WebWasher ICAP server

  The ACNS 5.5.1 software allows you to download large files over an ICAP-enabled network. Because of some compatibility issues with the data trickling feature of the WebWasher ICAP server, downloading large files over ICAP-enabled ACNS networks used to fail before the download was completed. However, starting with release 5.5.1, the ACNS software has become compatible with the data trickling feature of the WebWasher ICAP server.

- Support for accessing blank pages over ICAP-enabled networks

  The ACNS 5.5.1 software allows you to access blank pages (null body pages, where there is no content) over an ICAP-enabled ACNS network. When you try to access blank pages, earlier versions of the ACNS software used to return error messages saying that the page could not be retrieved because the server was either unreachable or temporarily busy.

- New CLI command and GUI support for configuring the ICAP connection timeout

  The ACNS 5.5.1 software allows you to configure a timeout value for ICAP connections. You can configure an ICAP connection timeout value using the **icap connection-timeout** *minutes* global configuration command.

Configuring ICAP Settings

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

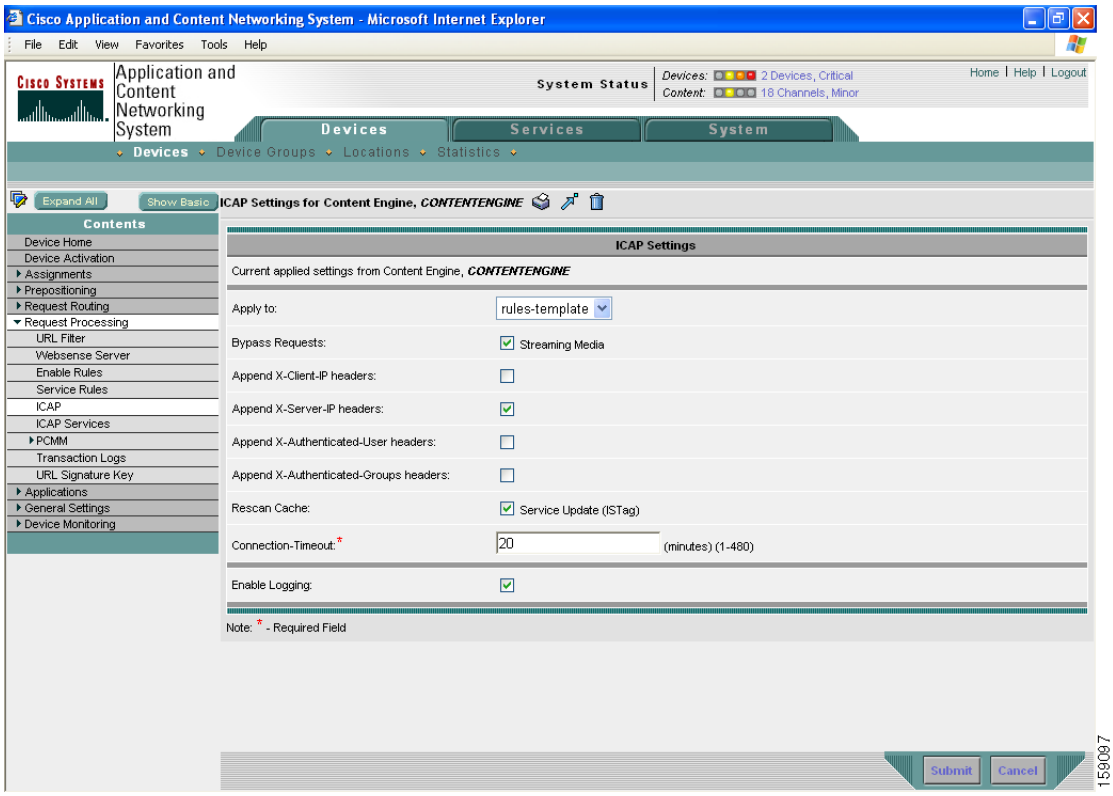
Step 1 Choose **Devices > Devices**. The Devices window appears.

Step 2 Click the **Edit** icon next to the Content Engine that you want to configure. The Device Home window appears.

Step 3 To display the entire table of contents, click the **Show All** button above the Contents pane.

Step 4 In the Contents pane, choose **Request Processing > ICAP**. The ICAP Settings window appears. (See Figure 16-1.) Table 16-1 describes the fields in this window and provides the corresponding CLI global configuration commands.

Figure 16-1 ICAP Settings Window



Step 5 From the Apply to 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 **rule action use-icap-service** rule defined in the Rules Template.

Step 6 If you do not want to bypass ICAP processing for streaming media requests (that is, requests from Windows, RealMedia, and QuickTime media players) that enter the Content Engine, uncheck the **Bypass Requests Streaming Media** check box. This check box is checked by default.

Requests from streaming media clients are by default bypassed for ICAP processing. Disable the **icap bypass streaming-media** option to force strict rechecking of the cached content every time the ISTag changes. The ISTag is a field in the HTTP response header that allows ICAP servers to send a service-specific cookie to an ICAP client, representing the current state of the sevice. The ISTag might change as a reslult of an update to the server version, to a virus-pattern-file, or to the policy.

You can configure the Content Engine to append the client and server IP address headers to requests that are passed to the ICAP server. The **icap append x-headers** option specifies that the ICAP extension headers are passed to the ICAP server during the session negotiation between the Content Engine and the ICAP server. This capability allows you to use your ICAP server to perform URL filtering based on the client IP address and server IP address.

Step 7 To add an X-client-IP header for ICAP processing of HTTP requests, check the **Append X-Client-IP headers** check box.

Step 8 To add an X-server-IP header for ICAP processing, check the **Append X-Server-IP headers** check box.

You can configure the Content Engine to append the username and groupname headers to the request that is passed to the ICAP server was added. This capability allows you to use your ICAP server to perform URL filtering based on username and groupname. The currently supported authentication schemes include LDAP, NTLM, RADIUS, and TACACS+.

Step 9 To allow the username information to be passed to the ICAP server for global services, check the **Append X-Authenticated-User headers** check box. This check box is unchecked by default. When this check box is checked, the x-authenticated-user information is inserted into the ICAP request that is sent to the ICAP server.

Step 10 To allow the groupname information to be passed to the ICAP server for global services, check the **Append X-Authenticated-User-Groups headers** check box. This check box is unchecked by default. When check box is checked, the x-authenticated-groups information is inserted into the ICAP request that is sent to the ICAP server.

Step 11 To rescan cached objects, check the **Rescan Cache Service Update (ISTag)** check box. The ISTag is a field in the HTTP response header that can change as a result of such things as updates to the ICAP server or ICAP policy. A change in the ISTag triggers the ICAP server to send a cookie to the ICAP client, indicating that a change has occurred and that the client should not use any previously cached material and should instead rescan the cache.

Step 12 To configure the ICAP connection timeout interval, enter a number between 1 and 480 in the Connection-Timeout field. The default is 20 minutes.

The **icap connection timeout** option allows you to stop an ICAP transaction that might be blocking traffic for longer than the specified time. This feature allows you to control the use of system resources and any additional overhead to the system that might be caused by a blocked transaction.

Step 13 To configure and enable logging in ICAP standard logging format for ICAP exchanges between ICAP servers and Content Engines, check the **Enable Logging** check box.

Step 14 To save your settings, click **Submit**. A "Click Submit to Save" message appears in red next to the current settings line when there are pending changes to be saved after you have applied default and device group settings. 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.

Table 16-1 ICAP Settings

GUI Parameter	Function	CLI Command
Apply to	The **all** option applies ICAP processing to all HTTP requests. The **rules-template** option applies ICAP processing only to requests that match the **rule action use-icap-service** rule defined in the Rules Template.	**icap apply** {**all** | **rules-template**}
Bypass Requests Streaming Media	When checked, bypasses ICAP processing for streaming media requests.	**icap bypass streaming-media**
Append X-Client-IP headers	Appends the client IP address header to requests that are passed to the ICAP server.	**icap append-x-headers x-client-ip**
Append X-Server-IP headers	Appends the server IP address header to requests that are passed to the ICAP server.	**icap append-x-headers x-server-ip**

Table 16-1 ICAP Settings (continued)

GUI Parameter	Function	CLI Command
Append X-Authenticated-User headers	Allows the username information to be passed to the ICAP server for global services.	**icap append x-authenticated-user**
Append X-Authenticated-Groups headers	Allows the groupname information to be passed to the ICAP server for global services.	**icap append x-authenticated-groups**
Rescan Cache	Rescans cached objects.	**icap rescan-cache ISTag-change**
Connection-Timeout	Adjusts the timeout interval (in minutes) for ICAP server connections.	**icap connection-timeout** *minutes*
Enable Logging	Enables logging in ICAP standard logging format for ICAP exchanges between ICAP servers and Content Engines.	**icap logging** {**format** {**standard**} | **enable**}

Configuring ICAP Services

ACNS 5.5 software supports the following three vectoring points:

- Request modification-precache

  This vectoring point applies to requests from clients. In the case of a cache miss, 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.

- Request modification-postcache

  This vectoring point applies only to requests after a cache miss and before the request is 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 vectoring 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 vectoring points (especially request modification-precache) may become overloaded with HTTP requests because all requests pass through this point. We 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 vectoring point. An ICAP service configured at a vectoring point can have only one load-balancing scheme, irrespective of the number of servers. However, multiple ICAP services that are configured at one or all of the vectoring points can have different load-balancing schemes.

Note When you use the Aggregate Settings option, you cannot use the ICAP Services for Content Engine window to modify or delete ICAP services that have been previously configured for device groups to which the Content Engine belongs. 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 > Devices**. The Devices window appears.

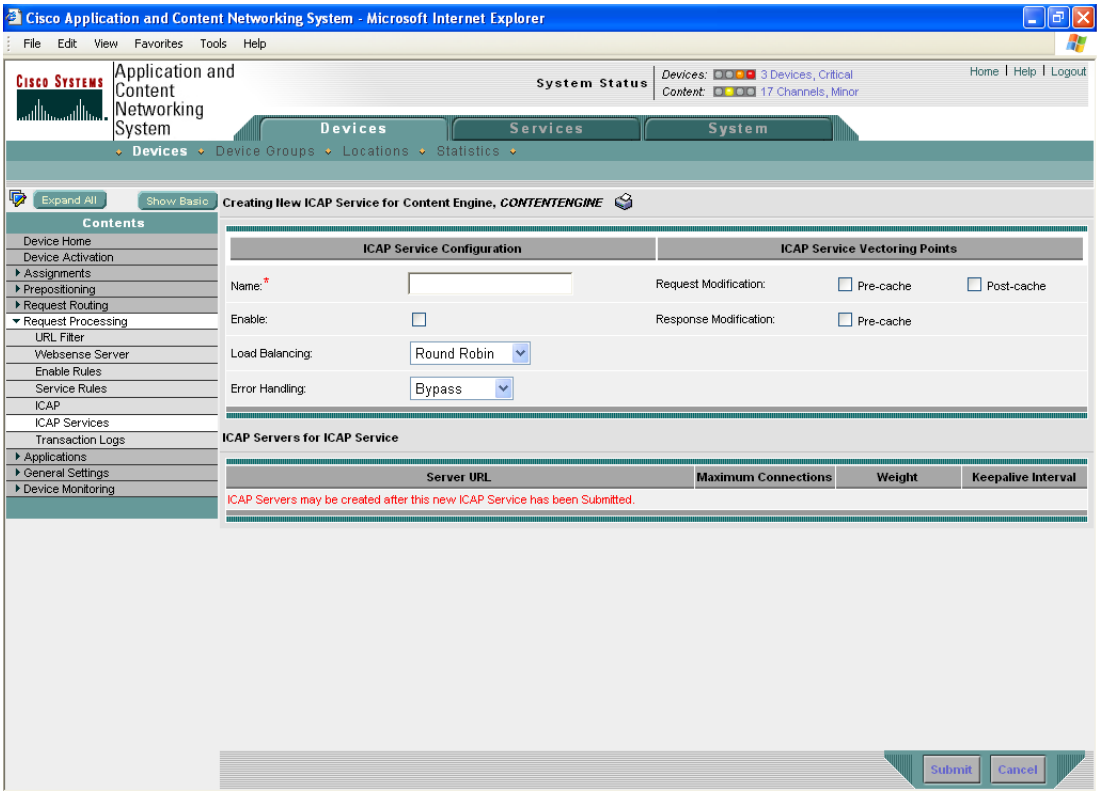
Step 2 Click the **Edit** icon next to the Content Engine for which you want to configure an ICAP service. The Device Home window appears.

Step 3 In the Contents pane, choose **Request Processing > 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. To display only the settings defined for the Content Engine, click the **No** radio button.

Step 5 In the taskbar, click the **Create New ICAP Service** icon. The Creating New ICAP Service for Content Engine window appears. (See Figure 16-2.) Table 16-3 describes the fields in this window and provides the corresponding CLI global configuration commands.

Figure 16-2 Creating New ICAP Service



Step 6 Configure the ICAP service, as follows:

a. In the Name field, enter a string that identifies the ICAP service to be configured.

b. To enable ICAP service, check the **Enable** check box.

c. From the Load Balancing drop-down list, choose the type of load-balancing mechanism for ICAP processing. Table 16-2 describes the load-balancing options.

Table 16-2 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 From the Error Handling drop-down list. choose the type of error handling mechanism for ICAP processing. 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:

a. To enable the vectoring point that is to be invoked when a Content Engine receives a request from a client, check the **Pre-cache** check box for request modification.

b. 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 **Post-cache** check box for request modification.

c. To enable the vectoring point that is to be invoked only when the response is from the origin server, check the **Pre-cache** check box for response modification.

Step 9 To save your settings, click **Submit**. 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.

Table 16-3 ICAP Service Settings

GUI Parameter	Function	CLI Command
ICAP Service Configuration		
Name	Identifies the ICAP service to be configured.	**icap service** *service-id*
Enable	Enables the ICAP service.	**icap service** *service-id* **enable**
Load Balancing	Configures the type of load-balancing mechanism for ICAP processing.	**icap service** *service-id* **load-balancing** {**round-robin** | **client-ip-hash** | **server-ip-hash** | **weighted-load**}

Table 16-3 ICAP Service Settings (continued)

GUI Parameter	Function	CLI Command
Error Handling	Configures the type of error handling mechanism for ICAP processing.	**icap service** *service-id* **error-handling** {**bypass** | **return-error**}
ICAP Service Vectoring Points		
Request Modification: Pre-cache	When checked, enables the vectoring point that is to be invoked when a Content Engine receives a request from a client.	**icap service** *service_id* **vector-point reqmod-precache**
Post-cache	When checked, enables 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.	**icap service** *service_id* **vector-point reqmod-postcache**
Response Modification: Pre-cache	When checked, enables the vectoring point that is to be invoked only when the response is from the origin server.	**icap service** *service_id* **vector-point respmod-precache**

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 vectoring 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 > Devices**. The Devices window appears.

Step 2 Click the **Edit** icon next to the desired Content Engine. The Content Engine Device Home window appears.

Step 3 In the Contents pane, choose **Request Processing > ICAP Services**. The ICAP Services for Content Engine window appears.

Step 4 The **Yes** radio button for aggregate settings is selected by default, so the ICAP services defined for the Content Engine and the device groups to which the Content Engine belongs are displayed. To display only the settings defined for the Content Engine, click the **No** radio button.

Step 5 To configure an ICAP service, click the **Edit ICAP Service** icon next to the ICAP server for which you wish to configure an ICAP service. 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. Table 16-4 describes the fields in this window and provides the corresponding CLI global configuration commands.

Step 7 In the Server Host field, enter the host name or IP address of the ICAP server.

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 the 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 that will be used for the ICAP service by using the URL format icap://ICAPserverIPaddress/service-name. The service name entered here must be supported by the ICAP vendor.

Step 10 In the Maximum Connections field, enter the maximum number of simultaneous connections that can be made to this server. 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. Do not leave this field blank if you chose the **Weighted** load-balancing method.

Step 12 In the Keepalive Interval field, specify the keepalive probe interval (in seconds) for this ICAP server. The default is 60 seconds.

Step 13 To save your settings, click **Submit**.

Table 16-4 ICAP Server for ICAP Service Settings

GUI Parameter	Function	CLI Command
Server Host	Host name or IP address of the ICAP server.	—
Server Port	Port number on which the ICAP server is to be configured to process HTTP requests.	—
Server Service Name	Path to the ICAP server configured on the Content Engine. Use the URL format: icap://ICAPserverIPaddress:port/service-name. The service name entered here must be supported by the ICAP vendor.	**icap service server server** *server_url*
Maximum Connections	Maximum number of simultaneous connections that can be made to this server.	**icap service server server** *server_url* **max-connections** *num*

Table 16-4 ICAP Server for ICAP Service Settings (continued)

GUI Parameter	Function	CLI Command
Weight	Percentage of load that can be redirected to this ICAP server.	**icap service server server** *server-url* **weight** *number*
Keepalive Interval	Keepalive probe interval (in seconds) for this ICAP server	**icap service server server** *server_url* **keepalive-interval** *seconds*

About ICAP Processing and Java Applets

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, and it overloads the ICAP server.

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

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

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

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

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

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

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

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

Displaying Information About an ICAP Configuration

To display the current ICAP configuration for a Content Engine, use the **show icap** EXEC command. The command output shows the status of the enabled ICAP features, the service definitions, a list of vectoring points, and an ordered list of ICAP services.

To display the definition and status of a specific ICAP service that is configured on the Content Engine, use the **show icap service** *service-name* EXEC command.

To display an ordered list of configured ICAP services and their status, use the **show icap vector-point** *vector-point-name* EXEC command.

```
ContentEngine# show icap vector-point ?
  reqmod-postcache  Display reqmod-postcache information
  reqmod-precache   Display reqmod-precache information
  respmod-precache  Display rspmod-precache information
```

Displaying Statistics for ICAP Services

To display ICAP statistics for all of the configured ICAP services, enter the **show statistics icap** EXEC command. This command has no arguments or keywords. There is no default behavior or values.

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.

Note The processing time on the Content Engine is directly related to the number of service rules configured. Processing times increase with an increase in the total number of rules configured. If the Content Engine processing time is greater than twice the datafeed poll rate, then the device goes offline until the processing is completed. You can avoid having the device go temporarily offline during processing by configuring a higher datafeed poll rate. The recommended datafeed poll rate for 750 service rules is 300 seconds. To configure the datafeed poll rate, see the "Modifying System Default Properties" section on page 20-18.

You can configure service rules by 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 5.5*.

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

- Enabling Rule Settings, page 16-14
- Configuring Service Rules, page 16-15

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 > Devices**. The Devices window appears.

Step 2 Click the **Edit** icon next to the Content Engine for which you want to enable rule settings. The Content Engine Device Home window appears.

Step 3 In the Contents pane, choose **Request Processing > Enable Rules**. The Service Rule Settings window appears. Check the **Enable** check box to enable the use of rule settings.

Step 4 To save the settings, click **Submit**.

To enable rules using the CLI, use the **rule enable** global configuration command.

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 for which you want to configure pattern lists. The Contents pane appears on the left.

Step 3 In the Contents pane, choose **Request Processing > Service Rules**.

Step 4 In the taskbar, click the **Create New Service Rules** icon. The Creating New Service Rules window appears.

Step 5 Configure a pattern list and add a pattern to it.

a. From the Rule Type drop-down list, choose **pattern-list**.

b. In the Rule Parameters field, configure the pattern list number and the pattern type, following the rules usage guidelines shown in the GUI. (See Table 16-5 for a description of pattern types.)

For example, to create pattern list number 72 with the pattern type *domain* and the yahoo.com domain as the domain to be acted on by an action, enter **72 domain yahoo.com** in the Rule Parameters field. (See Figure 16-3.)

Figure 16-3 Creating a New Pattern List and Defining a Pattern

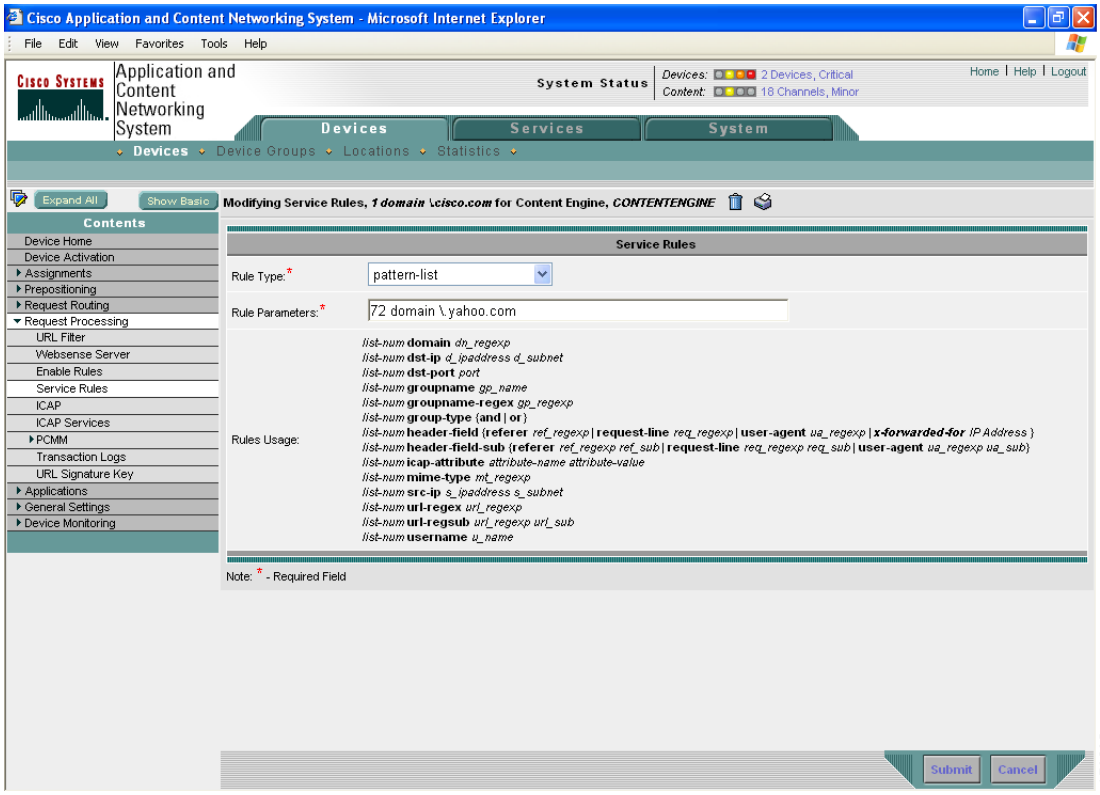


Table 16-5 Pattern Types

Pattern Type	Description	CLI Command
domain	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$" matches any domain name that ends with the ".foo.com" substring. In regular expression syntax, the dollar sign "$" metacharacter directs that a match is made only when the pattern is found at the end of a line.	**rule pattern-list** *list_num* **domain** *dn_regexp*
dst-ip	Matches the request's destination IP address and netmask against the specified 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.	**rule pattern-list** *list_num* **dst-ip** *d_ipaddress d_subnet*
dst-port	Matches the request's destination port number against the specified destination port number. Specify a port number.	**rule pattern-list** *list_num* **dst-port** *port*
groupname	Matches the groupname of the end user (web client that is requesting content) who was authenticated through LDAP or NTLM. This pattern can be applied only to request authentication for users who have been authenticated through LDAP or NTLM. Supports exact string comparison. The groupname comparison is case insensitive. Maximum length of groupname is 255 characters. Valid characters are an underscore and alphanumeric characters. If the groupname configuration in the Rules Template and the group name-based access list match, then the access list takes precedence. **Tip** If you intend to use the groupname pattern, make sure that you set the correct number of maximum group entries in the authentication group cache (the **http authentication cache max-group-entries** *number* global configuration command). This number should correspond to the maximum number of groups that could be returned during authorization queries (for example, the total number of groups defined on the AAA server.) The number can be from 500 to 12000. The number of entries in the authentication group cache is dependent on the physical resources available on the Content Engine.	**rule pattern-list** *list_num* **groupname** *name*

Table 16-5 Pattern Types (continued)

Pattern Type	Description	CLI Command
groupname-regex	Matches the group-name of the end user (web client that is requesting content) against a regular expression. Specify a regular expression. For example, use to configure a regular expression-based policy on groupnames or to OR multiple groupnames in the same line of in a single pattern list. For example, to OR three groupnames enter this command: `ContentEngine(config)# rule pattern-list 1 groupname-regex Engingeering|Marketing|Finance\` The specified action takes effect if any of the above groupnames matched the one in the request.	**rule pattern-list** *list_num* **groupname-regex** *group_name_regexp*
group-type	Specifies whether the pattern list is an AND or OR type. The default is OR.	**rule pattern-list** *list_num* **group-type** {**and** | **or**}
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. Requst header field pattern **x-forwarded-for** is supported for the **use-server** rule action only. This pattern is matched against the x-forwarded-for IP address in the HTTP header. The use-server action sends server-style HTTP requests from the Content Engine to the specified IP address and port on a cache miss.	**rule pattern-list** *list_num* **header-field** {**referer** *ref_regexp* | **request-line** *req_regexp* | **user-agent** *ua_regexp* | **x-forwarded-for** *ipaddress*}
header-field-sub	Request header field subpattern and substitute replacement pattern.	**rule pattern-list** *list_num* **header-field-sub** {**referer** *ref_regexp ref_sub* | **request-line** *req_regexp req_sub* | **user-agent** *ua_regexp ua_sub*}
icap-attribute	Specifies the attribute and value pair of the ICAP service.	**rule pattern-list** *list_num* **icap-attribute** *icap_attribute icap_value*
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.”	**rule pattern-list** *list_num* **mime-type** *mt_regexp*
scr-ip	Matches the request’s source IP address and netmask. Specify an IP address and a netmask.	**rule pattern-list** *list_num* **src-ip** *s_ipaddress s_subnet*
url-regex	Matches the URL against a regular expression. The match is case insensitive. Specify a regular expression.	**rule pattern-list** *list_num* **url-regex** *url_regexp*

Table 16-5 Pattern Types (continued)

Pattern Type	Description	CLI Command
url-regsub	For the **rewrite** and **redirect** actions, matches the URL against a regular expression to form a new URL in accordance with the pattern substitution specification. The match is case insensitive. The valid substitution index range is from 1 to 9.	**rule pattern-list** *list_num* **url-regsub** *url_regexp url_sub*
username	Matches the username of the end user (the web client that is requesting content) who was authenticated through LDAP, NTLM, RADIUS, or TACACS+. Specify the username or usernames. Maximum length of username is 255 characters for LDAP, RADIUS, or TACACS+ authentication. See below for information about the maximum length of usernames for NTLM authentication. Valid characters are an underscore and alphanumeric characters. The match supports exact string comparison. The username comparison is case insensitive. To specify multiple usernames in the same line for the same pattern list use a delimiter, as shown in this example: `ContentEngine(config)# rule pattern-list 1 username jdoe8,dsmith7,jsmith50` By default, the match does not consider the domain name, and matches for username only. To include domain name as well as username in the match, specify *domainname**username*, as shown in this example: `ContentEngine(config)# rule pattern-list 1 username domain cisco\jdoe8` For NTLM authentication, the domain\username:password:NTLM string must be 50 characters or less. If this string is greater than 50 characters, the domain name is truncated and the rule username pattern is not matched. An error message is generated in the system log in this situation. To match all users in a particular domain, enter this command: `ContentEngine(config)# rule pattern-list 1 username domain` *domainname*`\*` where *domainname* is the name of the domain (for example, cisco).	**rule pattern-list** *list_num* **username** *user_name*

Step 6 To save the settings, 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 16-6 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 16-4.)

Figure 16-4 Associating an Action with an Existing Pattern List

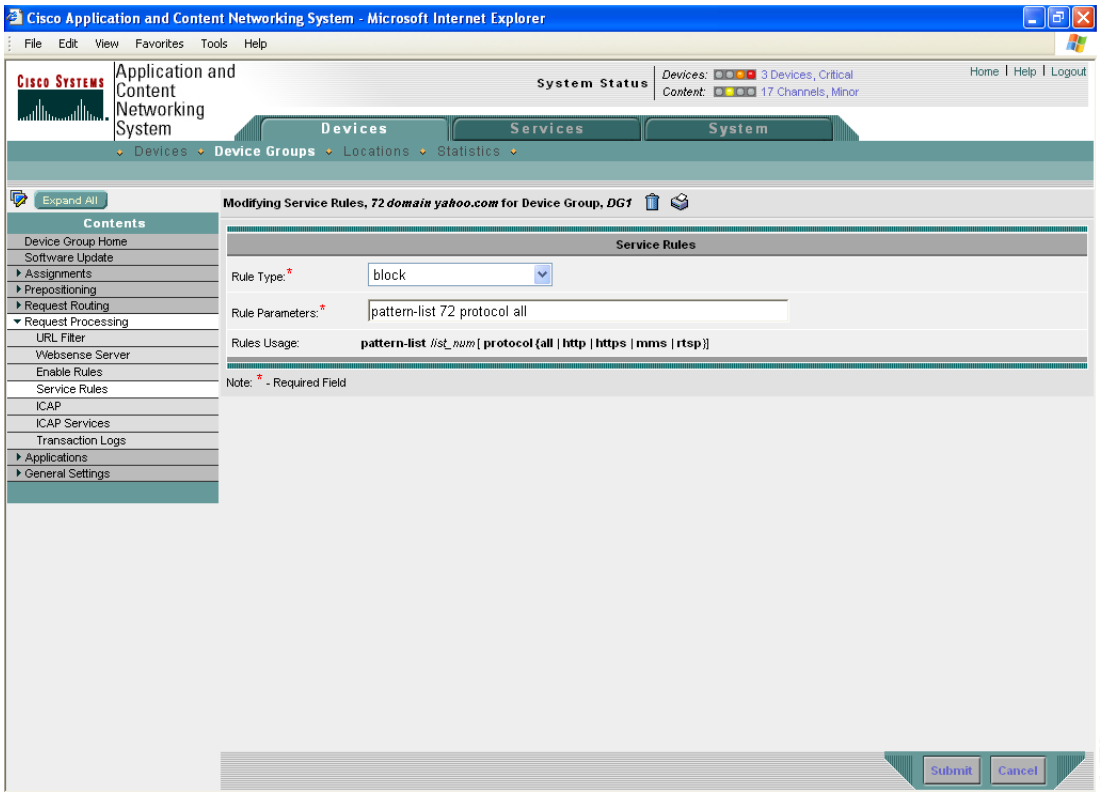


Table 16-6 Rule Actions

Action Type	Description	CLI Command
allow	Allows incoming requests that match the pattern list. This rule action can be used in combination with reset or block actions to allow selective types of requests. Allow does not carry any meaning as a stadalone action.	**rule action allow pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]
append-username-header	Appends the username in the request sent to the origin server.	**rule action append-username-header pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
block	Blocks this request and allows all others.	**rule action block pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]

Table 16-6 Rule Actions (continued)

Action Type	Description	CLI Command
cache-non-cacheable	Overrides the HTTP response headers and caches the objects. This rule action caches objects only if they are not authenticated. For authenticated objects, some origin servers do not send the Last-Modified and ETag entity headers, and revalidation of these objects cannot be performed by the Content Engine. These authenticated objects are served only from the origin server. If the server *does* send the Last-Modified and ETag headers, then these objects can be cached.	**rule action cache-non-cacheable ttl** {**day** *days* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}] | **hours** *hours* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}] | **minutes** *minutes* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}] | **seconds** *seconds* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]}
cache-only	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. If one or more rules specify this action, an object is cached 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.	**rule action cache-only pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]

Table 16-6 Rule Actions (continued)

Action Type	Description	CLI Command
dscp	Configures the IP ToS or DSCP code point field. 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. 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. Setting the Type of Service (ToS) or differentiated services code point (DSCP) is called packet marking, which allows 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. You can set specific ToS or DSCP values for the following: • Requests from the Content Engine to the server • Responses to the client on a cache hit • Responses to the client on a cache miss The ToS or DSCP can be set based on any of the policies matching the **src-ip** *s_ipaddress s_subnet*, **dst-ip** *d_ipaddress d_subnet*, **dst-port** *port*, **domain** *LINE*, **url-regex** *LINE*, or **mime-type** *LINE* options. In addition, you can now configure global ToS or DSCP settings with the **ip dscp** command. 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.	**rule action dscp client cache-hit** {**match-server pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}] | **set-dscp** *dscpvalue* | **set-tos** *tosvalue*} **rule action dscp client cache-miss** {**match-server pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}] | **set-dscp** *dscpvalue* | **set-tos** *tosvalue*}
freshness-factor	Determines the Time To Live if the request URL matches a specified regular expression. The **refresh** configuration takes priority over **freshness-factor** configurations.	**rule action freshness-factor** *exp_time* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
generate-url-signature	Generates the URL signatures in the Windows Media metafile response associated with pre-positioned content, based on the Content Engine configuration for the url-signature and this rule action.	**rule action generate-url-signature** [**include-client-src-ip**] **key-id-owner** *owner_num* **key-id-number** *id_num* **pattern-list** *list_num* [**protocol** {**all** | **http**}]
insert-no-cache	Inserts a no-cache header in the response.	**rule action insert-no-cache pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
no-auth	Does not authenticate.	**rule action no-auth pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]

Table 16-6 Rule Actions (continued)

Action Type	Description	CLI Command
no-cache	Does not cache this object. If both the **no-cache** and selective-cache actions are matched, **no-cache** takes precedence.	**rule action no-cache pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
no-proxy	For a cache miss, does not use the configured upstream proxy but contacts the server directly.	**rule action no-proxy pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
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.	**rule action redirect** *url* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]
redirect-url-for-cdn	Redirects the original request to a specified URL for the ACNS network.	**rule action redirect-url-for-cdn pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]
refresh	For a cache hit, forces an object freshness check with the server.	**rule action refresh pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
reset	Issues a TCP RST. This reset request is useful when resetting Code Red or Nimda virus requests.	**rule action reset pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]
rewrite	Rewrites the original request as a specified URL.	**rule action rewrite pattern-list** *list_num* [**protocol** {**all** | **http** | **https** | **rtsp**}]
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 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.	**rule action use-dns-server** {*hostname* | *ip-address*} **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
use-icap-service	Applies ICAP processing and uses a specific ICAP service only for 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.	**rule action use-icap-service** *service-name* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
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.	**rule action use-proxy** {*hostname* | *ip-address*} *port* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
use-server	Sends server-style HTTP requests from the Content Engine to the specified IP address and port on a cache miss.	**rule action use-server** {*hostname* | *ip-address*} *port* **pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]

Table 16-6 Rule Actions (continued)

Action Type	Description	CLI Command
use-xforward-clt-ip	Uses client IP address in the forwarded header for filtering.	**rule action use-xforward-clt-ip pattern-list** *list_num* [**protocol** {**all** | **http** | **https**}]
validate-url-signature	Validates the URL signature for a request using the configuration on your Content Engine for the url-signature and allows the request processing to proceed for this request.	**rule action validate-url-signature error-redirect-url** *url* **pattern-list** *list_num* [**protocol** {**all** | **http** | **rtsp**}]

Step 8 To save the settings, click **Submit**.

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, page 16-24
- Adding a Pattern to an Existing Pattern List, page 16-25
- Associating an Action with an Existing Pattern List, page 16-25

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#
```

The **rule action use-proxy** *proxy* **pattern-list** *number* global configuration command can be used to configure only one proxy for a particular pattern list. If the use-proxy feature is configured without failover (for example, you have entered the **rule action use-proxy 10.16.0.0 8080 pattern-list 1**

command), the Content Engine will send the request to the use-proxy (the server in this example with the IP address 10.16.0.0). If the Content Engine does not obtain a response from the use-proxy, then it will send an error message to the client without failing over to the HTTP outgoing proxy.

If the use-proxy feature is configured with failover (for example, you have entered the **rule action use-proxy 10.16.0.0 8080 failover pattern-list 1** command), the Content Engine will send the request to the use-proxy (in this example, the server with the IP address of 10.16.0.0). If the Content Engine does not obtain a response from the use-proxy, then it fails over to the specified HTTP outgoing proxy.

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 with 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 get an immediate return on investment by increasing productivity and recapturing 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.
- Permit access only to URLs specified in a list.
- Direct traffic to an N2H2 server for filtering.
- Direct traffic to a Websense enterprise server for filtering.

For information about configuring the Websense software, go to the following website: http://www.websense.com/content/home.aspx

- 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.

Configuring URL Filter Settings Using the Content Distribution Manager GUI

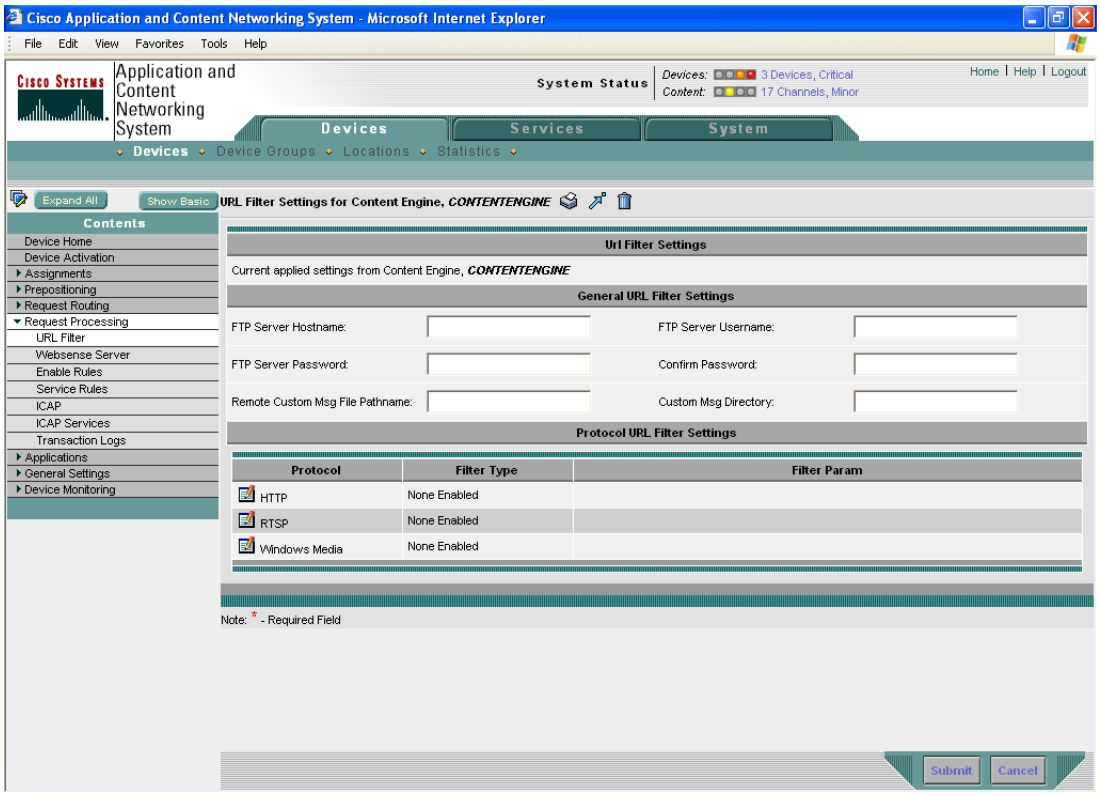
To configure URL filter settings for the Content Engine, follow these steps:

Step 1 From the Content Distribution Manager GUI, choose **Devices > Devices**. The Devices 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 In the Contents pane, choose **Request Processing > URL Filter**. The URL Filter Settings window appears. (See Figure 16-5.)

Figure 16-5 URL Filter Settings Window



Step 4 Enter information for the general URL filter settings. (See Table 16-7 for descriptions of the filter-setting parameters.)

Step 5 To create custom blocking messages, do the following:

a. In the Remote Custom Msg File Pathname field, enter the path name for the remote file that contains the custom message directory.

b. In the Custom Msg Directory field, enter the directory name that contains the *block.html* file. (For more information, see the "Configuring Custom Blocking Messages Using the CLI" section on page 16-32.)

Step 6 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 16-7.)

Step 7 To save the settings, click **Submit**.

Table 16-7 URL Filter Settings Window

GUI Parameter	Function	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.	—

Table 16-7 URL Filter Settings Window (continued)

GUI Parameter	Function	CLI Command
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.	—
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 *block.html*.	**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 the 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 the remote good site file.	—
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*

Table 16-7 URL Filter Settings Window (continued)

GUI Parameter	Function	CLI Command
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.	**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* | *hostname*} **port** *1-65535* **timeout**
Enable Websense Filtering	Enables the use of a Websense server for URL filtering.	**url-filter http websense enable**
Embedded	Enables 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 enable**
WebSense Server Hostname	Host name or IP address of an external Websense server	**url-filter http websense server** *hostname*
WebSense Port	Port number on which the Websense server is accepting requests.	**url-filter http websense server** {*ipaddress* | *hostname*} **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**

Table 16-7 URL Filter Settings Window (continued)

GUI Parameter	Function	CLI Command
WebSense Request Timeout	Number of seconds the Content Engine should wait for a response from the Websense server.	**url-filter http websense server** {*ipaddress* | *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 are certain that a different value is required.	**url-filter http websense server** {*ipaddress* | *hostname*} **port** *1-65535* **timeout** *seconds* **connections** *1-250*

Configuring URL Filters Using the CLI

Note Replace HTTP with either RTSP or WMT in the examples that follow if you want to use URL lists on these protocols. Make sure the URLs that are either accessed or denied match the protocol used in the command.

Use the **url-filter wmt** global configuration commands to configure local list URL filtering for WMT requests. If you configure an RTSP URL in a WMT bad sites list, then it blocks both the RTSPT and RTSPU URLs, as well as the RTSP URL that is specified in the bad sites list.

To deny requests for specific HTTP URLs using the CLI, follow these steps:

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

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

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

Note We recommend creating a separate directory under local1 to hold the bad and good lists, for example, /local1/filtered_urls.

Step 3 Point to the badurl.lst file by entering the following command:

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

Step 4 Deny the bad URLs by entering the following command:

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

Use the **no** form of this command to disable URL blocking as follows:

```
ContentEngine(config)# no url-filter http bad-sites-deny enable
```

To permit specific HTTP URLs to the exclusion of all other URLs using the CLI, follow these steps:

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

In this file, enter the HTTP URLs that you want to exclusively allow. The list of URLs in the goodurl.lst file must be written in the form 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 bad and good sites lists, for example, /local1/filtered_urls.

Step 3 Point to the goodurl.lst file by entering the following command:

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

Step 4 Permit only the good URLs by entering the following command:

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

Use the **no** form of this command to disable the allowance of good URLs as follows:

```
ContentEngine(config)# no url-filter http good-sites-allow enable
```

Configuring Custom Blocking Messages Using the CLI

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

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

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.

Creating a Text File URL List for URL Filtering

You can configure the Content Engine to deny client requests for URLs that are listed in a badurl.lst file, or you can 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 RTSP.

Note The local list file for each protocol should not contain URLs that belong to other protocols. For example, the HTTP local list file should contain only HTTP, HTTPS, or FTP 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, the device issues a warning 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. You must write the list of URLs in the badurl.lst file in the form http://www.domain.com/ 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. You must write the list of URLs in the goodurl.lst file in the form http://www.domain.com/ delimited with carriage returns.

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

Note We recommend that you create 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 for each 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**).

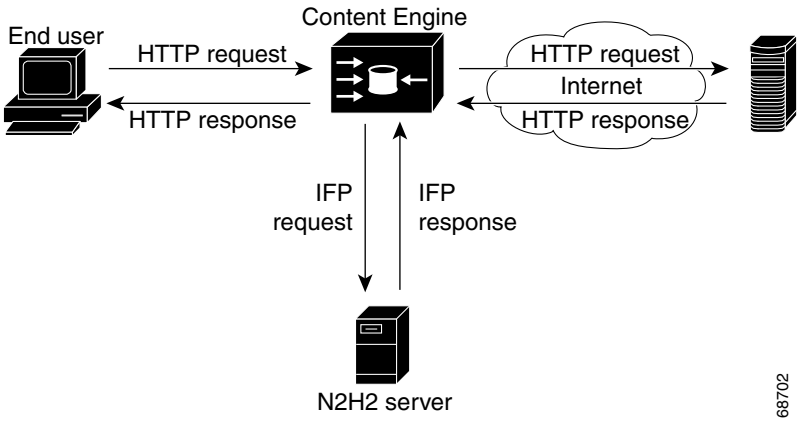
Configuring URL Filtering with the N2H2 Server

Note URL filtering with the N2H2 server applies only 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 by using the N2H2 server. (See Figure 16-6.) 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 16-6 N2H2 Filtering



URL filtering that uses an N2H2 server is applied to HTTP, FTP, or HTTPS traffic before the service rule mechanism is applied, regardless of whether or not the requested object is in the cache. 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 16-8 lists the N2H2 features supported by the Content Engine. One N2H2 server can support multiple Content Engines simultaneously.

Table 16-8 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.

Enabling N2H2 Filtering Using the Content Distribution Manager GUI

To configure N2H2 filter settings for the Content Engine, follow these steps:

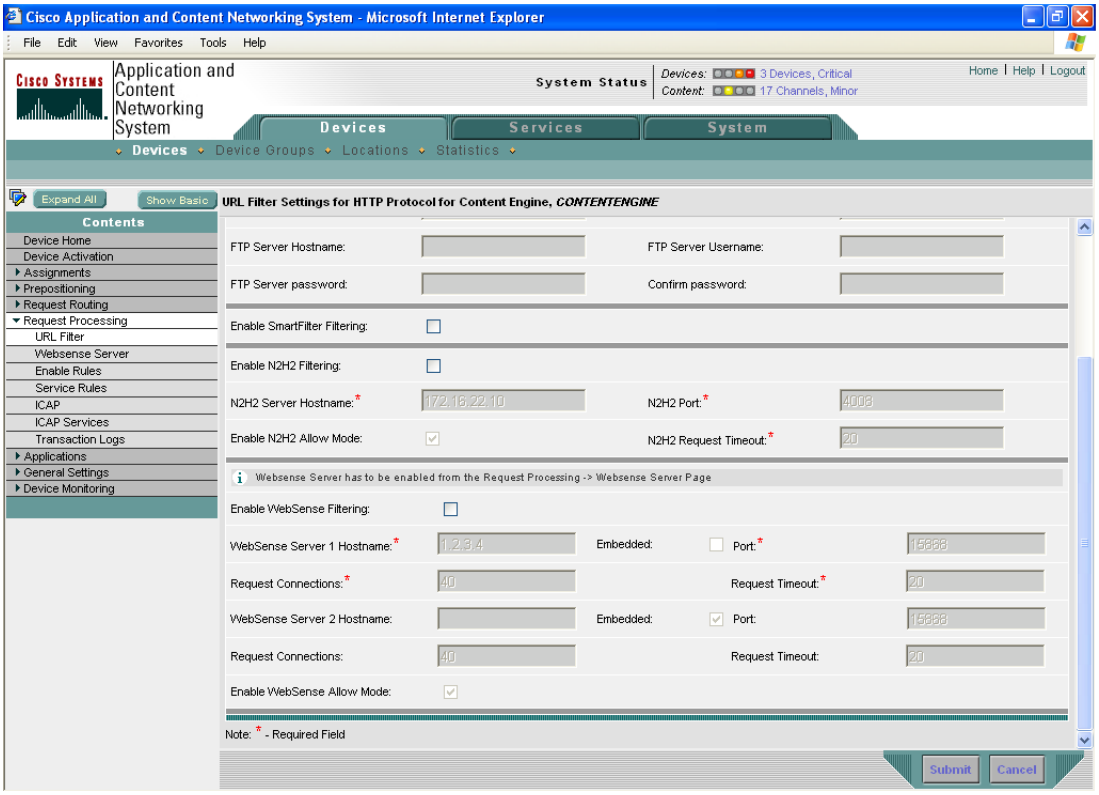
Step 1 From the Content Distribution Manager GUI, choose **Devices > Devices**. The Devices window appears.

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 Choose **Request Processing > URL Filter**. The URL Filter Settings window appears. (See Figure 16-5.)

Step 4 Under the Protocol URL Filter Settings heading, click the **Edit Filter** icon next to the HTTP protocol. The URL Filter Settings for HTTP Protocol window appears. (See Figure 16-7.) Table 16-9 describes the fields in this window and provides the corresponding CLI global configuration commands.

Figure 16-7 URL Filter Settings—HTTP URL Filter Settings



Step 5 To save the N2H2 settings, click **Submit**.

Table 16-9 URL Filtering Using the N2H2 Server

GUI Parameter	Function	CLI Command
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**

Table 16-9 URL Filtering Using the N2H2 Server (continued)

GUI Parameter	Function	CLI Command
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.	**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**

Enabling N2H2 Filtering Using the CLI

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

Step 1 Display the URL filtering schemes that are currently enabled on this Content Engine for requests over HTTP by entering the following command:

```
Content Engine# show url-filter http
URL filtering is DISABLED

Local list configurations
=================================
Good-list file name :
Bad-list file name : /local1/badfile.txt
Custom message directory :

Websense server configuration
=================================
Websense server IP     : 2.42.0.144
Websense server port   : 15868
Websense server timeout: 20 (in seconds)
Websense server connections: 40
Websense server IP     : <none>
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       : <none>
N2H2 server port     : 4005
N2H2 server timeout  : 5 (in seconds)
N2H2 allow mode is ENABLED
```

Step 2 Make sure that no other URL filtering scheme (for example, Websense or SmartFilter software) is currently enabled for requests over HTTP.

Only one URL filtering scheme per protocol can be active at a time.

Step 3 Configure the Content Engine to use an external N2H2 server for URL filtering by entering the following command:

```
ContentEngine(config)# url-filter http N2H2 server {[hostname | ip-address]} [port portnum
[timeout seconds]]
```

where

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

In the following example, the Content Engine is configured to use an N2H2 server that has an IP address of 172.16.22.10. The Content Engine will send IFP requests to this N2H2 server on port 4008 and will wait for up to 20 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 20
```

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

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

Step 4 Enable the N2H2 URL filtering scheme on this Content Engine by entering the following command:

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

If the URL filter is already enabled with N2H2 or other filtering schemes, the **enable** command fails. Even if the server IP address is not configured, the command is accepted, but filtering does not take effect until the N2H2 server is properly configured. To ensure that all traffic is filtered by the N2H2 server with a cluster of Content Engines, make sure to use the **url-filter http N2H2 enable** command on each Content Engine in the cluster.

Step 5 Allow HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) to pass through when the N2H2 server is enabled but the Content Engine has problems communicating with the N2H2 server by entering the following command:

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

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 Display the request-reply statistics for the communication between the Content Engine and the N2H2 server by entering the following command:

```
ContentEngine(config)# show statistics url-filter http N2H2
N2H2 URL Filtering Statistics:
             Lookup requests transmitted = 0
                Lookup response received = 0
                      Requests timed out = 0
                   Number of retransmits = 0

                Requests BLOCKed by N2H2 = 0
                   Requests OKed by N2H2 = 0

Allow Mode Statistics:
                 No available connection = 0
           Error sending lookup requests = 0
          Error recving 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 = 0
                       Avg response time = 0
                   Socket update count = 0
```

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

You can clear the statistics by entering 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 statistics counters are reset to 0.

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

N2H2 Configuration and Restrictions

Only one URL filtering scheme can be active for each protocol. 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 (or the GUI equivalent) 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 the N2H2 server uses to listen 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.

Configuring URL Filtering with Websense Enterprise Software

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

Websense Enterprise is an Internet filtering software that transparently monitors, reports on, and manages employee use of the Internet. Cisco ACNS 5.5 software supports Websense Enterprise software version 5.5 on all Cisco Content Engine platforms.

Note The integrated Websense Enterprise software version 5.5 in ACNS 5.5 software requires a minimum of 512 MB of RAM. We recommend that you upgrade the RAM on your device to 512 MB or greater, or move your integrated Websense server to to another device that has at least 512 MB of RAM.

You can configure your ACNS network to use an external enterprise server with Websense Enterprise software installed, or you can enable the Content Engine integrated Websense server, or you can configure both an external and an integrated Websense server for the Content Engine. An external Websense server communicates with the Content Engine over the network, whereas an integrated Websense server runs internally to the Content Engine. Communications between the Content Engine caching processes and an integrated Websense server happen internally to the Content Engine.

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

http://www.websense.com/content/KnowledgeBase.aspx

Websense integration documents are listed under the heading, “Integration-specific Installation Guides.”

Websense 6.2 Support

The ACNS 5.5.5 release supports URL filtering using an external Websense server with Websense Version 6.2. To configure your Content Engines for Websense URL filtering in a centrally managed deployment, see the “Using a Websense Enterprise Sever” section. To configure your Content Engines for Websense URL filtering in a locally managed deployment, see the “Configuring Standalone Content Engines for Websense URL Filtering” section.

Note Websense Version 6.2 is not supported as an integrated feature in the ACNS 5.5.5 software.

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

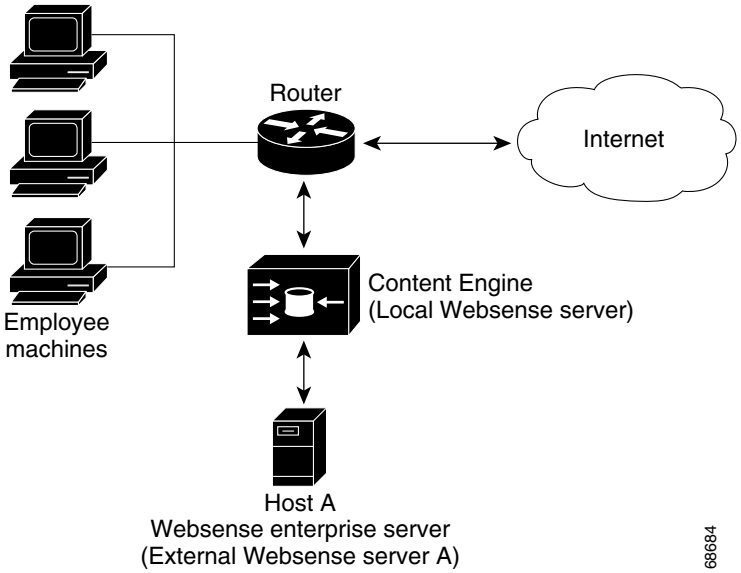
http://www.websense.com/content/home.aspx.

Using a Websense Enterprise Server

Figure 16-8 shows a network configuration that uses an external Websense enterprise server as a filtering engine for the Content Engine. The Content Engine enforces the filtering policy configured on the remote Websense server. Refer to the Websense documentation for further information on Websense filtering policies.

Note Clicking the **Save Changes** button from the Websense Enterprise Manager window in the Websense GUI does not save the Websense configuration across device reboots. You need to use the **write memory** command to save the Websense configuration changes across reboots.

Figure 16-8 URL Filtering with a Websense Server



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 times out all HTTP requests (HTTP, FTP-over-HTTP, or HTTPS-over-HTTP requests) and blocks and allows 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 reenable it.

In ACNS software Release 5.2 and later releases you can configure up to two Websense servers for failover purposes. The order in which you configure 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 16-10.

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

Step 1 Specify the necessary information about the external Websense server by entering the following command:

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

where

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

The following example shows how to configure a 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 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 (Optional) Configure a secondary Websense server for failover purposes as follows:

- To configure the integrated Websense server as the secondary Websense server, use the **url-filter http websense server local** global configuration command. (For more information about configuring the interated Websense server, see the "Using the Integrated Websense Server" section on page 16-43.)
- 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 Websense server running on Host B.)

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

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

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

Step 3 Enable Websense as the current URL filtering scheme for HTTP on this Content Engine by entering the following command:

```
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/content/home.aspx.

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

Using the Integrated Websense Server

You can also configure the Content Engines in your network to use the integrated Websense server. The integrated Websense server 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 on the Content Engine and the Websense URL database is downloaded the first time, CPU usage can be very high, so we recommend that you enable the 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 executables, as well as the configuration and logging files, are stored in the above 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 an amount that is greater than the default sysfs on your Content Engine.

Configuring Ports for the Integrated 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 port 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 blocked pages that are served by the Websense server in response to the redirected request.

- 55806—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. These diagnostics connect to this port.

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

Note The Websense server must to be disabled and then enabled to pick up the newly configured ports.

About Websense Server Failover

The Websense server failover 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 16-10 lists the supported Websense server failover configurations.

Table 16-10 Supported Websense Server Failover Configurations

Supported Configurations	Integrated Websense Server	Remote Websense Server
Option A	The integrated 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).

Table 16-10 Supported Websense Server Failover Configurations (continued)

Supported Configurations	Integrated Websense Server	Remote Websense Server
Option B	The integrated Websense server is acting as the primary Websense server.	The secondary Websense server is running on an external host.
Option C	The integrated 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. To configure Websense server failover using the Content Distribution Manager GUI, see the next section, "Enabling Websense Filtering and Configuring a Failover Scheme."

Enabling Websense Filtering and Configuring a Failover Scheme

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

Step 1 From the Content Distribution Manager GUI, choose **Devices > Devices**.

Step 2 Click the **Edit** icon next to the Content Engine that you want to view. The Content Engine Device Home window appears.

Step 3 In the Contents pane, choose **Request Processing > URL Filter**. The URL Filter Settings for Content Engine window appears. (See Figure 16-5.)

Step 4 Click the **Edit Filter** icon next to the HTTP protocol. The URL Filter Settings for HTTP Protocol window appears. (See Figure 16-7.) Table 16-11 describes the fields in this window and provides the corresponding CLI global configuration commands.

Step 5 To enable URL filtering using a Websense server, check the **Enable WebSense Filtering** check box.

Note You can specify up to two Websense servers. The first server specified becomes the primary Websense server, and the second server specified becomes the secondary Websense server. You can configure the integrated Websense server to be the primary server and an external Websense server to be the secondary server, or the reverse.

Step 6 To configure the Content Engine to use an external Websense server as the primary URL filtering server, enter the server IP address or hostname in the Websense Server 1 Hostname field.

Alternatively, to configure the Content Engine integrated Websense server as the primary URL filtering server, check the **Embedded** check box. The Websense Server Hostname field becomes unavailable. This option ensures that the URL filtering software uses the local Websense server and not a remote host as the Websense server.

Note When you configure the Content Engine to use the integrated Websense server, you must install the appropriate server components and enable the Websense server service on the Content Engine before any URL filtering can take place. (See the "Installing Websense Server Components for the Content Engine" section on page 16-53.)

Step 7 In the Port and Request Timeout fields, leave the default settings as they appear unless different settings are required.

Step 8 In the Request Connections field, leave the default setting at 40 connections for each CPU unless you are certain that a different value is required.

Step 9 To configure a secondary Websense server for the Content Engine, choose either an external server or the integrated server, and follow the same steps outlined for configuring the primary Websense server options.

Step 10 To enable HTTP access to a website if the Websense server does not respond, check the **Enable WebSense Allow Mode** check box.

Step 11 To confirm your settings, click **Submit**.

Table 16-11 Enabling Websense URL Filtering

GUI Parameter	Function	CLI Command
Enable WebSense Filtering	Makes the WebSense Server 1 and 2 Hostname and Port fields available.	**url-filter http websense server**
WebSense Server 1 Hostname	Enables the external Websense server specified by the IP address or name of the external Websense server.	**url-filter http websense server** *hostname*
Port	Port number on which the Websense server will accept requests from the Content Engine. The default port is 15868.	**url-filter http websense server** *hostname* **port** *port_num*
Request Timeout	Maximum amount of time that the Content Engine will wait for a response from the Websense server. The default is 20 seconds.	**url-filter http websense server** *hostname* **port** *port_num* **timeout** *num* **connections** *num*
Request Connections	Number of persistent connections to the Websense server per CPU.	

Table 16-11 Enabling Websense URL Filtering (continued)

GUI Parameter	Function	CLI Command
Embedded	Configures the Content Engine to send URL filtering requests to the integrated Websense server running on the Content Engine.	**url-filter http websense server local port** *port_num*
Enable WebSense Allow Mode	Allows HTTP access to a website if the Websense server does not respond. 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, however, the Content Engine blocks all traffic through the Content Engine. By default, **allowmode** is enabled.	**url-filter http websense allowmode enable**

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

Configuring Websense Server Failover and URL Filtering Using the CLI

In the following scenario, the Content Engine acts 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 will be used to activate the individual Websense services on the Content Engine as follows:

- 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 hostname or IP address, and its port number) on the Content Engine.

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

where

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

Either the local or the remote policy server must be running before you can activate any of the services of the local Websense server (the local EIM server, the local network agent, the local eDirectory agent, the local RADIUS agent, and the local user service) on the Content Engine. Local and remote policy server configuration are mutually exclusive.

Step 2 Activate the local EIM server on the Content Engine by entering the following command:

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

Step 3 Activate the local user service on the Content Engine by entering the following command:

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

Step 4 Activate the local network agent on the Content Engine by entering the following command:

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

Step 5 Activate the local eDirectory agent on the Content Engine by entering the following command:

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

Step 6 Activate the local RADIUS agent on the Content Engine by entering the following command:

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

Step 7 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 entering the following command:

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

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.

Note If you are using the local Websense server with a cluster of Content Engines, make sure that you enable the local Websense server on each Content Engine (enter the **websense-server enable** global configuration command on each Content Engine in the Content Engine cluster).

Step 8 Configure the Content Engine to use the local Websense server as the primary Websense server by entering the following command:

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

Note You can use the **url-filter http websense server** global configuration command to configure different settings (for example, the timeout, the 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 Websense server on port 15868, waits 20 seconds for a response from Websense server before timing out the connection, and establishes 40 persistent connections per CPU.

In the previous 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 as the primary Websense server for the Content Engine.

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

Step 9 Configure the Content Engine to use an external Websense server as the secondary Websense server by entering the following command:

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

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

In the previous 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 sends the requests to this secondary Websense server on port 4006, waits up to 90 seconds for a response from this server before timing out the connection, and establishes 90 persistent connections per CPU.

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

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

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

Step 10 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 as follows:

- 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 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 Websense server.

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

Step 11 Enable URL filtering on the Content Engine by entering the following command:

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

Step 12 Configure the default policy using the Websense Manager GUI. This step should be performed for both the local and the remote Websense server.

a. Use the Websense Manager GUI to add a policy server:

- Right-click the left pane of the Websense Manager main window.
- Choose **Add Policy Server**.
- In the displayed dialog box, enter the IP address of the Content Engine that is running the local (internal) Websense server.

b. Connect to the Websense policy server that is running on the Content Engine:

- In the left pane, double-click on the policy server (which could be the Content Engine IP address, for example).
- Enter the username and password, and then click **OK**.

c. Use the Websense Manager GUI to configure a Websense policy:

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

Note For more information about how to use the Websense Manager GUI, go to the following website: http://www.websense.com/content/home.aspx.

Note Clicking the **Save Changes** button from the Websense Enterprise Manager window does not save the Websense configuration modifications across device reboots. You need to use the **write memory** command to save the Websense configuration changes across reboots.

Step 13 Configure the HTTP proxy on the Content Engine by entering the following command:

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

Step 14 Display statistics for both the primary and the secondary Websense servers by entering the following command:

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

About Websense Server Components

ACNS 5.5 software supports the following Websense Enterprise software version 5.5.2 server components on the integrated Websense server:

- Policy Server
- EIM Server
- User Service
- Network Agent
- Radius Agent
- eDirectory Agent
- Logon Agent (ACNS support for this server component has been added in the ACNS 5.5 release.)

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

Table 16-12 describes these Websense 5.5.2 services.

Table 16-12 Integrated Websense 5.5.2 Server Components Supported in ACNS 5.5 Software

Name	Description
Policy Server	Hosts all of the policy information that you have configured through the external Websense Manager GUI. Communicates the policy information to the other components of the integrated Websense server. **Note** The integrated Policy Server or the specified remote Policy Server must be already installed and running before you can activate the EIM Server, User Service, Network Agent, RADIUS Agent, and eDirectory components. All of these five components need the IP address of the Policy Server for installation. With Websense 5.2 software, you can use the integrated Websense Policy Server or a remote Websense Policy Server to activate the individual server components on the Content Engine.
EIM Server	Provides the URL filtering functionality when used with proxy servers, firewalls, and caching appliances.
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 16-12 Integrated Websense 5.5.2 Server Components Supported in ACNS 5.5 Software

Name	Description
Network Agent	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: • Database applications, such as SQL Net • File transfer applications, such as FTP and Gopher • Instant messaging and chat applications, such as Yahoo Messenger, and MSN Messenger • Mail and collaborative tools, such as POP3, SMTP, and NetMeeting • Network operating system applications, such as Daytime, finger, NTP, SSH, and Telnet • Remote access applications, such as VNC and pcANYWHERE • Streaming media applications, such as RTSP, Windows Media, and Liquid Audio • Other (for example, Network News Transfer Protocol [NNTP])
RADIUS Agent	Enables URL filtering based on user-based or group-based policies for users who are authenticated through an external RADIUS server. This agent transparently identifies the users who access the network and are authenticated through the RADIUS authentication scheme. When the Content Engine is supplied with this information, the Content Engine can apply policies to users and groups of the users who access the network remotely. This agent acts as a proxy that forwards the RADIUS messages between the RADIUS client and the external RADIUS server. For the integrated RADIUS agent to work properly, you must have configured the RADIUS settings (for example, the IP address of the external RADIUS server) on the Content Engine.
eDirectory Agent	Enables URL filtering based on user-based or group-based policies for users who are authenticated through LDAP. This agent works in conjunction with the Novell eDirectory to transparently identify users who access the network and are authenticated through the LDAP authentication scheme. When the Content Engine is supplied with this information, the Websense filtering service can filter requests based on the policies applied to the users or groups. This agent uses LDAP to gather the user login session information from the Novell eDirectory, which authenticates users logging in to the network. This agent associates each authenticated user with the IP address. With the help of the Websense integrated user service, the integrated eDirectory agent supplies this information to the Websense filtering service. For this integrated eDirectory agent to work properly, you must have configured such settings as the administrative distinguished name.
Logon Agent	Enables URL filtering based on user-based or group-based policies for users as they log on to the network through a Windows client machine. This agent's associated logon application captures logon sessions as users log on to Windows domains in a network. The Logon Agent communicates with the Websense User Service to provide up-to-date user logon session information to Websense for filtering purposes. The Logon Agent identifies users in a real-time manner as they log on to a domain, whereas a Domain Controller (DC) Agent identifies users by periodically querying domain controllers and workstations. Because the Logon Agent identifies users in a real-time manner, the Websense Filtering Service can accurately filter Internet access based on the policies assigned to particular users, groups, workstations, or networks. The Logon Agent, like the eDirectory agent, transparently identifies users. Support for the Logon Agent was added in the ACNS 5.4.1 software release.

Installing Websense Server Components for the Content Engine

To install Websense server components using the Content Distribution Manager GUI, follow these steps:

Step 1 Choose **Devices > Devices**. The Devices window appears, listing all the device types configured in the ACNS network.

Step 2 Click the **Edit** icon next to the name of the Content Engine for which you want to configure URL filter settings. The Device Home for Content Engine window appears.

Step 3 In the Contents pane, choose **Request Processing > Websense Server**. The Websense Server Settings for Content Engine window appears. Table 16-13 describes the fields in this window and lists the corresponding CLI commands.

Step 4 To install the integrated Websense Policy Server component on the Content Engine, check the **Install policy server component** check box. (See the "About Websense Server Components" section on page 16-51.)

Alternatively, to specify a remote Websense server that is running the Websense Policy Server component, check the **Specify remote policy server** check box. You must specify either local Content Engine installation of Policy Server, or specify a remote Policy Server, but not both.

If you specified a remote Policy Server, enter the IP address of the remote host in the Remote Host field. The Policy Server port field displays port 55806 and needs to be changed only if you need to configure a port other than 55806.

Step 5 To install the Websense server EIM Server component on the Content Engine, check the **Install EIM component** check box.

Step 6 To install the Websense server User Service component on the Content Engine, check the **Install User component** check box.

Step 7 To install the Websense server Network Agent component on the Content Engine, check the **Install Network Agent component** check box.

Step 8 To install the Websense server Logon Agent component on the Content Engine, check the **Install Logon Agent component** check box.

Step 9 To install the Websense server RADIUS Agent component on the Content Engine, check the **Install Radius Agent component** check box.

(For more information, see the "Specifying RADIUS Agent Settings for the Content Engine Using the CLI" section on page 16-54.)

Step 10 To install the Websense server eDirectory Agent component, check the I**nstall eDirectory Agent component** check box.

(For more information, see the "Specifying eDirectory Agent Settings for the Content Engine Using the CLI" section on page 16-55.)

Step 11 To configure an administrative password that the Content Engine should use to contact the external eDirectory Server to request a database search, enter a string in the Password field.

Step 12 To confirm the password, reenter the same string in the Confirm Password field.

Step 13 To enable the Websense server and activate the installed components, check the **Enable Websense Server** check box.

Step 14 To confirm your settings, click **Submit**.

Table 16-13 Websense Server Settings for the Content Engine

GUI Parameter	Function	CLI Command
Install policy server locally	Installs the integrated policy server on the Content Engine.	**websense-server service policy local activate**
Specify remote policy server	Activates GUI fields for specifying the remote host IP address and port number.	—
Remote Host	Specifies the remote policy server to be used to activate the other integrated server components on the Content Engine.	**websense-server service policy remote host** *remote-policy-server IP-address* **port** *remote-policy-server port-number*
Port	Specifies the port number of the remote Policy Server. The default port number is 55806.	
Install EIM component	Installs the integrated EIM server on the Content Engine.	**websense-server service eim activate**
Install User component	Installs the integrated user service on the Content Engine. Use the **no** form of this command to deactivate it.	**websense-server service user activate**
Install Network Agent component	Installs the integrated network agent on the Content Engine.	**websense-server service network-agent activate**
Install Logon Agent component	Installs the integrated logon agent on the Content Engine. The Logon Agent uses port 15880.	**websense-server service logon-agent activate**
Install Radius Agent component	Installs the integrated RADIUS agent on the Content Engine.	**websense-server service radius-agent activate**
Install eDirectory Agent component	Activates the integrated eDirectory agent on the Content Engine.	**websense-server service edir-agent activate**
Password / Confirm Password	Administrative password that the Content Engine should use to contact the external eDirectory Server to request a database search.	**websense-server service edirectory-agent edir-server administrative-passwd** *password*
Enable Websense Server	Activates the Websense server components that you installed.	**websense-server enable**

Specifying RADIUS Agent Settings for the Content Engine Using the CLI

In the Websense GUI Manager versions 5.5 and 6.1 that are supported by the Websense 5.5.2 software, you can configure the RADIUS agent through the GUI, as well as through CLI commands. Consequently, in the ACNS 5.4.1 software release, all of the global configuration CLI commands that were related to configuring the RADIUS agent have been removed from the ACNS CLI command set. However, the CLI command that is used to active the RADIUS agent has been retained.

The following global configuration commands for configuring the RADIUS agent were removed in the ACNS 5.4.1 software release:

websense-server service radius-agent incoming [**auth-port** *port number*] [**accnt-port** *port number*]

websense-server service radius-agent outgoing [**host** *remote-RADIUS-server IP-address*] [**auth-port** *port number*] [**accnt-port** *port number*]

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

Specifying eDirectory Agent Settings for the Content Engine Using the CLI

In the Websense GUI Manager versions 5.5 and 6.1 that are supported by the Websense 5.5.2 software, you can configure the eDirectory agent through the GUI, as well as through CLI commands. Consequently, in the ACNS 5.4.1 software release, all of the global configuration CLI commands that were related to configuring the eDirectory agent have been removed from the ACNS CLI command set. However, the CLI command that is used to active the eDirectory agent has been retained.

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

websense-server service edirectory-agent edir-server administrative-passwd *password*

The following global configuration command for configuring the eDirectory agent was removed in the ACNS 5.4.1 software release:

websense-server service edir-agent edir-server [**administrative-dn** *administrative-distinguished-name*] [**host** *remote-eDirectory-server IP-address*] [**root-context** *root-context*]

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

Websense Status and Statistics Commands

Additional CLI commands to use with Websense URL filtering are as follows:

- **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.

  The **show url-filter http** command also 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 displays 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#
```

- **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.

- **show statistics url-filter http websense**

  This 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 by using the **clear statistics url-filter http websense**, and **clear statistics all** commands. All the statistics counters are then reset to 0.

- **write memory**

  This EXEC command saves Websense configuration files (websense.init and ws.cfg) modified from the external Websense Manager GUI across disk reconfiguration and ACNS software release upgrades (which might erase disk content).

  You must enter this command to have the most recent configuration modifications, including websense.ini file modifications and the Websense URL filtering configuration changes.

  If you do not use the **write memory** command before a 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.

Websense Configuration and Restrictions

Only one URL filtering scheme can be active per protocol. To enable Websense URL filtering, 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 by using the **url-filter http websense server** *IP address* [**port** *1-65535*] [**timeout** *1-120*] command (or the GUI equivalent) and enabling the Websense server with the **url-filter http websense enable** command (or the GUI equivalent).

The server IP address and port number that are configured in the Content Engine must match the IP address of the Websense server and the port on which the Websense server listens to filter 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.

Configuring URL Filtering with SmartFilter Software

Note URL filtering with SmartFilter software applies only to HTTP, FTP, or HTTPS protocols.

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 that is 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 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. These GUI components download configurations into the Content Engine to be used by the SmartFilter process.

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

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

- When you upgrade or downgrade the Content Engine to a different release of the ACNS software, if there is a difference in the SmartFilter plug-in version, the SmartFilter database and configuration files are deleted and default configurations are loaded. This change occurs because the configuration details might be changed with each new version of the SmartFilter software. After each upgrade or downgrade of the SmartFilter plug-in, a fresh database has to be downloaded from the SmartFilter Administration Console to the Content Engine.

Note The ACNS 5.2.x, ACNS 5.3.x, 5.4.x, 5.5.1 and later software releases support the SmartFilter software version 4.0.1. The ACNS 5.5.5 and later software support the SmartFilter software version 4.1.

- If the Content Engine is deployed behind a firewall, you must use the **external-ip** *external-ip-address* global configuration command to configure the Content Engine's external IP address; otherwise, the advanced block page feature may not work properly.
- The Smartfilter software and the cache process on the Content Engine will be restarted in the following situations:
  - If you configure the Content Engine's external IP address when the Smartfilter software is running on the Content Engine
  - If the Content Engine's external IP address is not configured and you change the Content Engine's interface IP address when the Smartfilter software is running on the Content Engine.
- Port 9014 is reserved for Smartfilter Advanced block pages.

Note To obtain a copy of the Smartfilter authentication server software and information about configuring the SmartFilter software, go to the following website: http://www.securecomputing.com.

SmartFilter Version 4.1 Integration

The plug-in for SmartFilter software Version 4.1 has been integrated with the ACNS 5.5.5 software. This version of SmartFilter has the following features:

- Listens on port 9014 for block page requests.

  The ACNS caching process listens on port 9015 for blocked pages and forwards the packets to port 9014 where SmartFilter listens.
- Provides advanced block pages with a new look and feel.
- Provides a new feature called temporary user override. (See the "About the Temporary User Override Feature" section on page 16-59.)

Note To configure SmartFilter Version 4.1 features, you must obtain the SmartFilter Administration Console version 4.1.1, which can be downloaded from the Secure Computing website.

About the Temporary User Override Feature

In the ACNS 5.5.5 software release, the temporary user override is a new feature that is available with the SmartFilter software version 4.1. This feature allows the SmartFilter plugin to override the filtering mechanism that is being applied to the user group to which users have been added in the authentication server (in Step 3). You must configure this new feature through the SmartFilter Administrator Console (version 4.1.1).

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

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

Step 2 From the SmartFilter Administrator Console, add the authentication server.

Step 3 From the SmartFilter Administrator Console, add the users to the authentication server.

Step 4 Deploy the changes to the authentication server.

Step 5 From the SmartFilter Administrator Console, select the Content Engine, and add the configured authentication server to the Content Engine's list of authentication servers.

Step 6 From the SmartFilter Administrator Console, add the users who should be allowed to override the filtering mechanism in the overrides for the Content Engine.

Step 7 Deploy the changes to the Content Engine.

About the SmartFilter Control List

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

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

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

SmartFilter Software and the Action No-Auth Command Rule Interaction

The **rule action no-auth** global configuration command permits specific login and content requests to bypass authentication and authorization features, such as LDAP, RADIUS, NTLM, or TACACS+. In the following example, any requests from the source IP address (src-ip) 172.16.53.88 are not authenticated:

```
ContentEngine(config)# rule enable
ContentEngine(config)# rule action no-auth pattern-list 1 protocol all
ContentEngine(config)# rule pattern-list 1 src-ip 172.16.53.88 255.255.255.255
```

If the software is configured for authentication and SmartFilter URL filtering, requests that are allowed to bypass authentication will also bypass the SmartFilter URL filter.

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

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

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

To support this new feature, the following CLI changes were made:

- The **rule** global configuration command supports a new action called the **no-url-filtering** action.

  The **no-url-filtering** action supports the following rule patterns: src-ip, dst-ip, dst-port, domain, group-name, groupname-regex, header-field, url-regex, and username.

  Note Patterns can be ANDed or ORed by using the group-type pattern (for example, **rule pattern-list 1 group-type and**). The default is OR.

- The output of the **show run**, **show statistics rule all**, and **clear statistics rule all** EXEC commands now includes information about the new **no-url-filtering** action.
- The **show statistics rule http action no-url-filtering** EXEC command was added to enable you to display statistics for the **no-url-filtering** action.

The following is an example of how you can use this new bypass URL filtering feature with Websense URL filtering. First, the **rule action no-url-filtering** command is specified and then associated with a specific pattern list (pattern list 100). Next, the **domain** pattern type is added to pattern list 100 in order to configure the Content Engine to match requests that have “foo.com” as the domain. In this scenario, Websense URL filtering has already been configured and enabled on the Content Engine.

```
ContentEngine (config)# rule action no-url-filtering pattern-list 100
ContentEngine (config)# rule pattern-list 100 domain .*foo.com
ContentEngine (config)# rule enable
```

When the Content Engine receives an HTTP or HTTPS request that has “foo.com” as the domain, the **rule action no-url-filtering** rule is matched. The Content Engine bypasses URL filtering for that particular request (as shown in the partial output of the **debug http proxy** command below).

```
Oct 28 12:25:12 Content Engine 3: Rule action no-url-filtering match - Bypassing
urlfiltering
```

If the **rule action no-url-filtering** rule is matched and SmartFilter URL filtering is being used instead of Websense URL filtering, the output of the **debug http proxy** command would be as follows:

```
Oct 28 12:25:12 Content Engine 3: Rule action no-url-filtering match - Bypassing
SmartFilter processing
```

When the Content Engine receives an HTTP or HTTPS request for websites other than “foo.com” (for requests that have “www.abc.com” as the domain), the **rule action no-url-filtering** rule is not matched. The Content Engine proceeds with Websense URL filtering for that particular request (as shown in the partial output of the **debug http proxy** command below).

```
Oct 28 12:28:06 Content Engine 3: Rule action no-url-filtering not hit - Proceed with
urlfiltering
```

If the **rule action no-url-filtering** rule is not matched and SmartFilter URL filtering is being used instead of Websense URL filtering, the output of the **debug http proxy** command would be as follows:

```
Oct 28 12:25:12 Content Engine 3: Rule action no-url-filtering not hit- Proceed with
SmartFilter processing
```

Execution Order of Rule Actions

In ACNS 5.2.3 software or later, the order in which the rule actions are executed is as follows:

1. Redirect-url-for-cdn (this action is only applicable for Content Engines that are registered with a Content Distribution Manager and is not applicable for standalone Content Engines)
2. No-auth (before authentication using RADIUS, LDAP, or NTLM)
3. Reset
4. Block / allow (See the Note that follows item 29.)
5. Redirect (before cache lookup)
6. Rewrite (before cache lookup)
7. No-url-filtering
8. Refresh (after cache lookup, in the case of cache hit)
9. Freshness-factor (after cache lookup, in the case of a cache hit)
10. Use-server
11. No-proxy
12. Use-proxy-failover
13. Use-proxy
14. Use-dns-server
15. ToS/DSCP server (TOS bits on the connection to the server)
16. ToS/DSCP client (TOS bits on the connection that the server uses to send response to client)
17. DSCP client cache-miss
18. DSCP client cache-hit
19. Insert-no-cache
20. No-cache

21. Cache (when the response is received from the server)
22. Selective-cache (when the response is received from the server)
23. Append-username-header
24. Use-icap-service
25. Use-xforward-clt-ip
26. No-persistent-connection
27. Cache-cookie
28. No-selective-cache
29. Allow

Note Allow and block carry the same precedence. The order of execution depends on the order of configuration between allow and block actions. Other actions always take precedence over allow. For example, a reset action always takes precedence over allow regardless of the order of configuration.

Configuring the Content Engine GUI for Secure or Nonsecure Access

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

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

Before you log in to the Content Engine GUI, make sure that you have the following information:

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

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

To configure the Content Engine GUI for secure or nonsecure access, follow these steps:

Step 1 Choose **Devices > Devices**. The Devices window appears, listing all the device types configured in the ACNS network.

Step 2 Click the **Edit** icon next to the Content Engine for which you want to enable the graphical user interface. The Device Home for Content Engine window appears.

Step 3 In the Contents pane, choose **Request Processing > GUI Server**. The GUI Server Settings for Content Engine window appears.

Step 4 Under the GUI Server Settings section, follow these steps:

a. To enable nonsecure access to the Content Engine GUI, check the **GUI Server Enable** check box.

b. In the GUI Server Port field, specify a port number for the graphical user interface server port. The default port for nonsecure access to the GUI is 8001.

Step 5 Under the Secured GUI Server Settings section, follow these steps:

a. To enable secure access to the Content Engine GUI, check the **Secured GUI Server Enable** check box.

b. In the Secured GUI Server Port field, specify a port number for the graphical user interface server port. The default port for secure access to the GUI is 8003.

Step 6 To save the settings, click **Submit**. A "Click Submit to Save" message appears in red next to the Current Settings line when there are pending changes to be saved after you have applied default and device group settings. You can also revert to the previously configured settings by clicking **Reset**. The **Reset** button is visible only when you have applied default or group settings to change the current device settings but have not yet submitted the changes.

If you try to leave this window without saving the modified settings, a warning dialog box prompts you to submit the changes. This dialog box appears only if you are using the Internet Explorer browser.

Note When secured access is enabled, unsecured access is automatically disabled.

Setting Up the Content Engine to Interoperate with Third-Party Policy Servers

Cisco ACNS 5.5 software allows you to set up interoperability with Camiant PCMM-compliant third-party policy servers to allocate guaranteed bandwidth for authorized requests of pre-positioned content.

Note The settings in this section are for a feature that is not released for general use. You should not set them unless you have been specifically instructed to do so by Cisco.

In a typical deployment scenario involving a third-party policy server for request authorization and allocation of guaranteed bandwidth, the request and response flow happens as follows:

1. A client logs in to a web portal, authenticates the user credentials, and requests a Windows Media file.
2. The portal inserts a cookie with necessary information about the user credentials for subsequent authorization of the request, and it directs the request to its back-end network, which is a content distribution network that runs Cisco ACNS software.
3. A Content Router in the back-end network picks up the request and redirects the message to the nearest available Content Engine that can serve the requested media file.

4. The Content Engine validates the requested URL based on the rule patterns defined on the Content Engine for authorization of the request. If the requested URL matches the rule pattern for domains that offer guaranteed bandwidth, the Content Engine passes the request to the third-party policy server, which handles authorization of the request and allocation of guaranteed bandwidth. (See the "Creating Rule Patterns and Rule Actions for Directing Requests to a Policy Server" section on page 16-67 for more information about rule patterns.)

Note The Content Engine uses ICAP protocol for communicating with the policy server for authorization of the request. You can set up an ICAP service for the policy server from the ICAP Services interface of the Content Distribution Manager to enable interoperability between your ACNS network and the policy server.

5. The entitlement server that is associated with the policy server verifies the user credentials. Based on the user credentials, the policy server returns one of the following messages to the Content Engine along with the requested URL:

Message Type	Explanation	Content Engine Action
OK with a Request ID	Valid user with privilege for guaranteed bandwidth.	Directs the streaming server to serve the content with guaranteed bandwidth.
OK	Valid user without privilege for guaranteed bandwidth.	Directs the streaming server to serve the content without guaranteed bandwidth.
Error	Valid user without privilege for receiving the requested content.	Drops the request.

6. The Content Engine, on receiving the OK with a Request ID message from the policy server, generates a URL signature and appends it to the requested URL, embeds the requested URL in an .asx file, and sends the .asx file back to the client. (See the "About URL Signature Keys" section on page 16-65.)

   The Content Engine provides two URLs to the client, one based on the RTSP protocol and another based on the MMS-over-HTTP protocol. The RTSP-based URL is the default URL, whereas the MMS-over-HTTP URL is for failover when the client is unable to access RTSP-based URLs. However, if both RTSP-based and MMS-over-HTTP URLs fail, the client has a third option to use progressive HTTP download. URL signatures are generated based on ACNS rule patterns set for specific domains and are used to ensure that the client is authorized to view the content.

7. After receiving the .asx file containing the URLs, the client contacts the Content Engine using one of the URLs based on the client settings. The Windows Media Server on the Content Engine validates the URL using the URL signature. If the signature passes validation, the Windows Media server begins streaming the content. If the client is using the MMS-over-HTTP protocol, then the HTTP proxy cache application on the Content Engine does the URL validation and content delivery.

8. Upon receiving the request from the client with the appropriate URL, the Content Engine issues an HTTP callout to the policy server to provide the IP address and port number of both the client and the server (Content Engine).

9. The policy server guarantees the required bandwidth for the stream between the two points, Content Engine port and client port, as specified in the HTTP callout. The Content Engine begins streaming the content.

10. When the stream ends, the Content Engine sends a teardown message to the policy server. Upon receiving the teardown request from the Content Engine, the policy server releases the allocated bandwidth for use by other streams.

About URL Signature Keys

URL signature keys are word values that Cisco ACNS software uses to ensure URL-level security. The URL signature key is a shared secret between the device that assigns the key and the device that decrypts the key. Based on your network settings, either the Content Engine itself or some other external device can assign the signature key to the URL, but the Content Engine decrypts the URL signature key.

Cisco ACNS software uses a combination of key owners, key ID numbers, and a word value to generate URL signature keys. To create request-specific URL signature keys, you can choose to append the IP address of the client that has made the request, to the URL signature key.

You can have a maximum of 32 key owners. Each key owner can have up to 16 key ID numbers. You can use a number between 1 and 32 for a key owner and a number between 1 and 16 for the key ID number. You can have only one key configured for a unique combination of key owner and key ID number at any given time. Key values can have a maximum of 16 characters (excluding double quotes at the beginning and end of the string).

The GUI allows quoted and unquoted strings. Double quotes (“”) are allowed at the beginning and end of the string only. If you do not surround the key string with double quotes, quotes will be added when you click Submit in the GUI.

Note The key supports using all special characters in the string, including space, pipe (|), question mark (?), forward slash (/), and back quote (‘).

URL signature key authentication is implemented by using the generate-url-signature and validate-url-signature rule actions that can be applied to specific rule patterns. (See the “Creating Rule Patterns and Rule Actions for Directing Requests to a Policy Server” section on page 16-67.)

Setting QoS Policy Server Settings for Content Engines

Cisco ACNS 5.5 software enables you to configure Content Engines to interoperate with Camiant PCMM-compliant third-party policy servers to allocate guaranteed bandwidth for authenticated requests of prepositioned content.

Note These settings are for a feature that is not released for general use. You should not set them unless you have been specifically instructed to do so by Cisco.

Note You must have a Camiant PCMM-compliant policy server in your network to use this feature.

To configure QoS Policy Server settings for a Content Engine or a device group, follow these steps:

Step 1 From the Content Distribution Manager GUI, choose **Devices > Devices** (or **Devices > Device Groups**). The Devices (or Device Groups) listing window appears.

Step 2 Click the **Edit** icon next to the device (or device group) for which you want to configure the QoS Policy Server settings. The Devices (or Device Group) home window appears.

Step 3 In the Contents pane, choose **Request Processing > PCMM > QoS Policy Service**. The QoS Policy Service Settings for Content Engine (or Device Group) window appears. Table 16-14 describes the fields in this window and provides the corresponding CLI commands.

Step 4 To enable policy server settings for the device (or device group), check the **Enable** check box.

Step 5 To specify the configuration file for the HTTP callout to the policy server, check the **Set Config File or URL** check box. The Config File or URL field becomes active.

The config file is an XML file that contains information on the callout URLs, attributes, application type, protocol, etc, for streaming the content.

Step 6 In the Config File or URL field, enter the filename or a URL from which the Content Engine can download the configuration file. This field supports only URLs that use the HTTP, HTTPS, or FTP protocols.

Step 7 To save the settings, click **Submit**.

Table 16-14 QoS Policy Service Settings

GUI Parameter	Function	CLI Command
Enable	Enables Camiant policy service.	**qos camiant-cdn-am-service enable**
Set Config File or URL	Allows you to specify the configuration file for the HTTP callout to the policy server.	**qos camiant-cdn-am-service config-file** *filename* or *URL*
Config File or URL	Filename or URL from which the Content Engine can download the configuration file.	

Creating New URL Signature Keys for Content Engines

Cisco ACNS 5.5 software enables you to create URL signature keys for unique identification of URL requests if your network is configured to use URL-level security features. The system allows you to define 32 different key owners and 16 keys for each key owner.

Note These settings are for a feature that is not released for general use. You should not set them unless you have been specifically instructed to do so by Cisco.

To create a URL signature key for a Content Engine or a device group, follow these steps:

Step 1 From the Content Distribution Manager GUI, choose **Devices > Devices** (or **Devices > Device Groups**). The Devices (or Device Group) listing window appears.

Step 2 Click the **Edit** icon next to the device (or device group) for which you want to configure the QoS Policy Server settings. The Device (or Device Group) home window appears.

Step 3 In the Contents pane, choose **Request Processing > PCMM > URL Signature Key**. The URL Signature Keys for Device (or Device Group) window appears.

Step 4 To create a new URL Signature Key, and click the **Create New URL Signature** icon in the taskbar. The Creating New URL Signature Key window appears. Table 16-15 describes the fields in this window and provides the corresponding CLI commands.

Alternatively, to go to the Edit URL Signature Key ID window, click the **Edit** icon corresponding to a URL Signature Key ID owner.

Step 5 In the Key ID Owner field, enter a number between 1 and 32 as the Key ID owner.

Step 6 In the Key ID field, enter a number between 1 and 16.

Step 7 In the URL Signature Key field, enter a unique URL signature key. This field does not support a space or the following special characters: pipe (|), question mark (?), double quotes (“), and back quote (‘).

Step 8 To save changes, click **Submit**.

Table 16-15 URL Signature Key Settings

GUI Parameter	Function	CLI Command
Key ID Owner	Specifies the ID number for the owner of this encryption key.	**url-signature key-id-owner** *owner_num*
Key ID	Specifies the encryption key ID number.	**url-signature key-id-owner** *owner_num* **key-id-number** *id_num*
URL Signature Key	Text of the encryption key. Maximum length is 16 characters.	**url-signature key-id-owner** *owner_num* **key-id-number** *id_num* **key** *encryption_key*

Creating Rule Patterns and Rule Actions for Directing Requests to a Policy Server

If your network is configured to work with third-party Camiant PCMM-compliant policy servers for servicing requests that require guaranteed bandwidth, you can use the following rule patterns and rule actions to filter the requests and to direct them to the policy server. The rule patterns and rule actions also enable you to generate URL signatures in the response for a valid request for a Windows Media metafile (.asx file extension) and to validate the URL signature on incoming requests to the Content Engine. (For more information on creating rule patterns and configuring rule actions, see the “Configuring Service Rules” section on page 16-14.)

The following rule patterns support the rule action **use-icap-service** for directing requests that require guaranteed bandwidth to the third-party policy server:

Rule Pattern	Description
url-regex	Filters the request based on any regular expression n the URL.
domain	Filters the request based on the domain name specified.
dst-ip	Filters the request based on the destination IP address.
dst-port	Filters the request based on the destination port number.
src-ip	Filters the request based on the IP address of the source.

Rule Pattern	Description
header-field user-agent	Filters the request based on the user agent specified in the request header.
header-field referer	Filters the request based on the referer in the request header.
header-field request-line	Filters the request based on the request line in the request header.

You can set the rule action **use-icap-service** for any of the rule patterns above. If the request matches the parameters that you have set for the rule pattern, then the Content Engine redirects the request to the third-party policy server using ICAP services. However, you must make sure that your network is configured to interoperate with the third-party policy server using ICAP service. You can set up the necessary ICAP configurations from the ICAP Server Settings window of Content Distribution Manager. Cisco ACNS software also provides a number of CLI commands to set up interoperability with a third-party policy server using ICAP services. For more information about CLI commands related to ICAP configuration for third-party policy servers, see the *Cisco ACNS Software Command Reference, Release 5.5* publication.

You can also use the rule pattern and rule action to generate URL signatures in the response for a valid request for a Windows Media metafile. You can use the following rule patterns to filter out requests for which you want to generate a URL signature key:

Rule Pattern	Description
url-regex	Filters the request based on any regular expression n the URL.
domain	Filters the request based on the domain name specified.
dst-ip	Filters the request based on the destination IP address.

For the rule patterns mentioned above, you can set the following rule actions:

Rule Action	Description
generate-url-signature	Generates the URL signatures in the Windows Media metafile response associated with pre-positioned content, based on the Content Engine configuration for the url-signature and this rule action.
validate-url-signature	Validates the URL signature for a request by using the configuration on your Content Engine for the url-signature and allows the request processing to proceed for this request