# Blue Coat® Systems
# SG Appliance

*Configuration and Management Guide*

*Volume 9: Access Logging*

*Version SGOS 5.1.x*

# Contact Information

Blue Coat Systems Inc.
420 North Mary Ave
Sunnyvale, CA 94085-4121

http://www.bluecoat.com/support/contact.html

bcs.info@bluecoat.com
http://www.bluecoat.com

For concerns or feedback about the documentation: documentation@bluecoat.com

Document Number: 231-02845
Document Revision: SGOS 5.1.x 03/2007

# Contents

# Chapter 1: About Access Logging

Access logging allows you to track Web usage for the entire network or specific information on user or department usage patterns. These logs and reports can be made available in real-time or on a scheduled basis.

**Note:** Event logging is not the same as access logging. *Event logging* allows you to specify the types of system events logged, the size of the event log, and to configure Syslog monitoring.

## Overview

SGOS can create access logs for the traffic flowing through the system; in fact, each protocol can create an access log record at the end of each transaction for that protocol (such as for each HTTP request).

**Note:** The only data that can be logged in an access log on the SG appliance are the access-log fields and the CPL fields (found in Appendix B: "Access Log Formats").

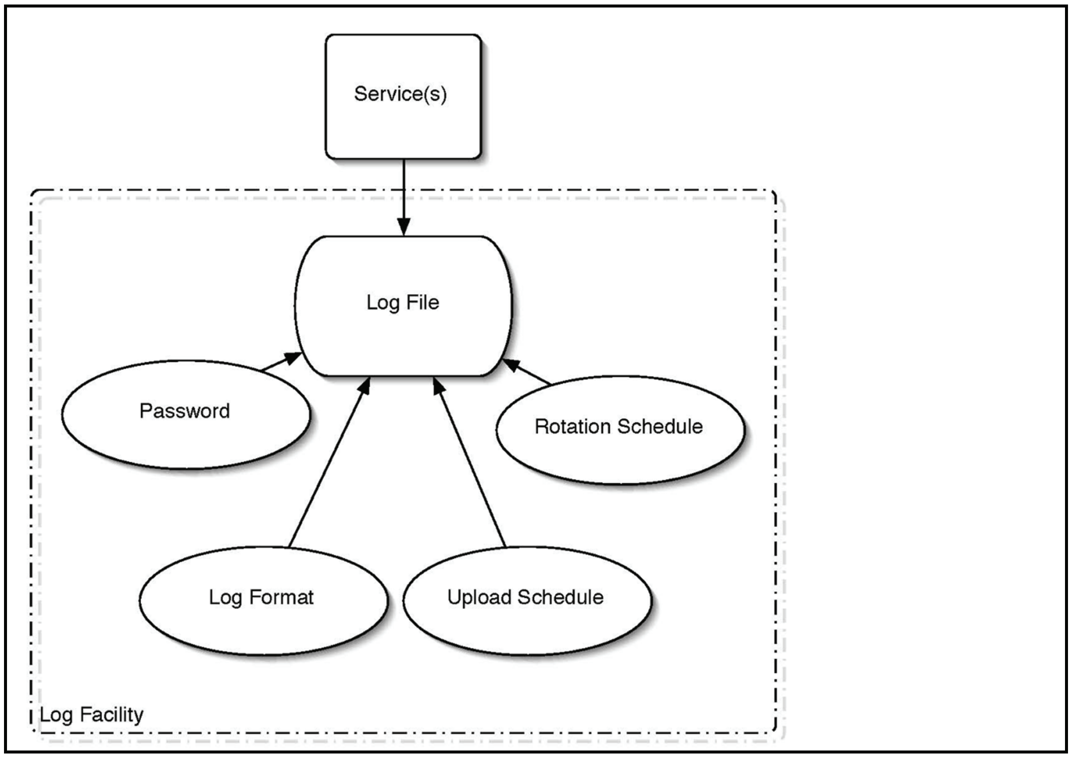
These log records can be directed to one or more log *facilities*, which associates the logs with their configured log formats, upload schedules, and other customizable components. In addition, access logs can be encrypted and digitally signed prior to upload.

Data stored in log facilities can be automatically uploaded to a remote location for analysis and archive purposes. The uploads can take placing using HTTP, FTP, or one of several proprietary protocols. Once uploaded, reporting tools such as Blue Coat Reporter can be used to analyze the log files. For information on using Blue Coat Reporter, refer to the *Blue Coat Reporter Configuration and Management Guide*.

## Understanding Facilities

A log facility is a separate log that contains a single logical file and supports a single log format. The facility contains the file's configuration and upload schedule information as well as other configurable information such as how often to rotate (switch to a new log) the logs at the destination, any passwords needed, and the point at which the facility can be uploaded.

Multiple access log facilities are supported, although each access log supports a single log format. You can log a single transaction to multiple log facilities through a global configuration setting for the protocol that can be modified on a per-transaction basis via policy.



## Understanding Protocols and Formats

The following protocols support configurable access logging:

- CIFS
- Endpoint Mapper
- FTP
- HTTP
- HTTPS Forward Proxy
- HTTPS Reverse Proxy
- ICP
- Instant Messaging
- Peer-to-peer (P2P)
- RealMedia/QuickTime
- SOCKS
- SSL
- TCP Tunnel
- Telnet
- Windows Media

SGOS can create access logs with any one of a number of log formats, and you can create additional types using custom or ELFF format strings. The log types supported are:

- NCSA common log format
- SQUID-compatible format
- ELFF (W3C Extended Log File Format)
- Custom, using the strings you enter
- SmartReporter, an ELFF log format compatible with the SmartFilter Reporter tool
- SurfControl, a log format compatible with the SurfControl Reporter tool
- Websense, a log format compatible with the Websense Reporter tool

The log facilities, each containing a single logical file and supporting a single log format, are managed by policy (created through VPM or CPL), which specifies the destination log format and log file.

## Enabling or Disabling Access Logging

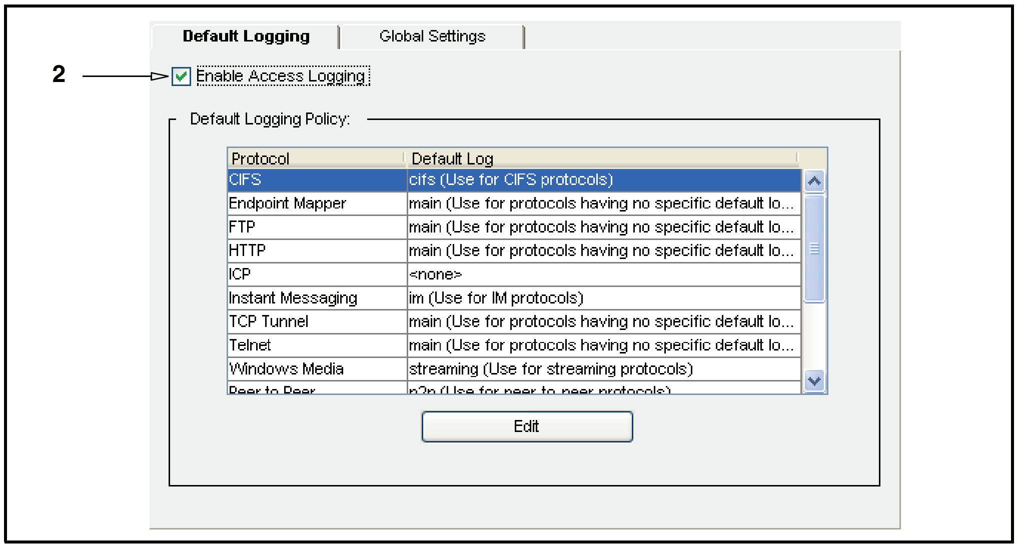
You can globally enable or disable access logging. If access logging is disabled, logging is turned off for all log objects, even if logging policy exists or logging configurations are set.

Once globally enabled, connection information is sent to the default log facility for the service. For example, HTTP traffic is logged to the main file.

By default, access logging is disabled on all new systems, but certain protocols are configured to use specific logs by default. When access logging is enabled, logging begins immediately for all configured protocols.

**To enable or disable access logging:**

1. Select **Configuration > Access Logging > General > Default Logging**.



[✓] Enable Access Logging

Default Logging Policy:

| Protocol | Default Log |
|---|---|
| CIFS | cifs (Use for CIFS protocols) |
| Endpoint Mapper | main (Use for protocols having no specific default lo... |
| FTP | main (Use for protocols having no specific default lo... |
| HTTP | main (Use for protocols having no specific default lo... |
| ICP | <none> |
| Instant Messaging | im (Use for IM protocols) |
| TCP Tunnel | main (Use for protocols having no specific default lo... |
| Telnet | main (Use for protocols having no specific default lo... |
| Windows Media | streaming (Use for streaming protocols) |
| Peer to Peer | p2p (Use for peer to peer protocols) |

2. Select **Enable** to enable access logging or deselect it to disable access logging.
3. Select **Apply** to commit the changes to the SG appliance.

*Volume 9: Access Logging* contains the following topics:

- Chapter 2: "Creating and Editing Log Formats" on page 9

- Chapter 3: "Creating and Editing Access Log Facility" on page 15
- Chapter 4: "Configuring the Upload Client" on page 21
- Chapter 5: "Configuring the Upload Schedule" on page 33
- Appendix A: "Glossary" on page 43
- Appendix B: "Access Log Formats" on page 51

## Document Conventions

The following section lists the typographical and Command Line Interface (CLI) syntax conventions used in this manual.

Table 1-1. Document Conventions

| Conventions | Definition |
|---|---|
| *Italics* | The first use of a new or Blue Coat-proprietary term. |
| `Courier font` | Command line text that appears on your administrator workstation. |
| *`Courier Italics`* | A command line variable that is to be substituted with a literal name or value pertaining to the appropriate facet of your network system. |
| **`Courier Boldface`** | A Blue Coat literal to be entered as shown. |
| { } | One of the parameters enclosed within the braces must be supplied |
| [ ] | An optional parameter or parameters. |
| \| | Either the parameter before or after the pipe character can or must be selected, but not both. |

# Chapter 2: Creating and Editing Log Formats

You should first decide what protocols and log formats you want to use, the logging policy, and the upload schedule. Then you can do the following:

- Associate a log format with the log facility.
- Associate a log facility with a protocol and/or create policies for protocol association and to manage the access logs and generate entries in them (if you do both, policy takes precedence).
- Determine the upload parameters for the log facility.

The Format tab allows you to create a format to use for your log facilities. Several log formats ship with the SGOS software, and they might be sufficient for your needs. If the formats that exist do not meet your needs, you can use the Format tab to create a custom or ELFF format and specify the string and other qualifiers used.

Several log formats already exist. For a description of each value in the log, see Appendix B: "Access Log Formats" on page 51.

- **cifs**: This is an ELFF format with the custom strings of

  ```
  date time c-ip r-ip r-port x-cifs-method x-cifs-server x-cifs-share
  x-cifs-path x-cifs-orig-path x-cifs-client-bytes-read x-cifs-server-
  bytes-read x-cifs-bytes-written s-action cs-username cs-auth-group
  s-ip
  ```

- **mapi**: This is an ELFF format with the custom strings of

  ```
  date time c-ip c-port r-ip r-port x-mapi-user x-mapi-method cs-bytes
  sr-bytes rs-bytes sc-bytes x-mapi-cs-rpc-count x-mapi-sr-rpc-count
  x-mapi-rs-rpc-count x-mapi-sc-rpc-count s-action cs-username cs-
  auth-group s-ip
  ```

- **im** (Instant Messaging): This is an ELFF format with the custom strings of:

  ```
  date time c-ip cs-username cs-auth-group cs-protocol x-im-method x-
  im-user-id x-im-user-name x-im-user-state x-im-client-info x-im-
  buddy-id x-im-buddy-name x-im-buddy-state x-im-chat-room-id x-im-
  chat-room-type x-im-chat-room-members x-im-message-text x-im-
  message-size x-im-message-route x-im-message-type x-im-file-path x-
  im-file-size s-action
  ```

- **main**: This is an ELFF format with custom strings of:

  ```
  date time time-taken c-ip sc-status s-action sc-bytes cs-bytes cs-
  method cs-uri-scheme cs-host cs-uri-port cs-uri-path cs-uri-query
  cs-username cs-auth-group s-hierarchy s-supplier-name rs(Content-
  Type) cs(User-Agent) sc-filter-result cs-category x-virus-id s-ip s-
  sitename
  ```

- **ncsa**: This is a reserved format that cannot be edited. The NCSA/Common format contains the following strings:

  ```
  remotehost rfc931 authuser [date] "request" status bytes
  ```

  The ELFF/custom access log format strings that represent the strings above are:

  ```
  $(c-ip) - $(cs-username) $(localtime) $(cs-request-line) $(sc-
  status) $(sc-bytes)
  ```

- **p2p**: This is an ELFF format with custom strings of:

  ```
  date time c-ip c-dns cs-username cs-auth-group cs-protocol x-p2p-client-type x-p2p-client-info x-p2p-client-bytes x-p2p-peer-bytes duration s-action
  ```

- **smartreporter**: This is a reserved format that cannot be edited. It contains the following string:

  ```
  localtime s-computername c-ip c-uri sc-filter-result cs-categories cs-user sc-bytes
  ```

- **squid**: This is a reserved format that cannot be edited. You can create a new SQUID log format using custom strings. The default SQUID format is SQUID-1.1 and SQUID-2 compatible.

  SQUID uses several definitions for its field formats:

  ```
  SQUID-1:time elapsed remotehost code/status/peerstatus bytes method URL
  ```

  ```
  SQUID-1.1: time elapsed remotehost code/status bytes method URL rfc931 peerstatus/peerhost type
  ```

  SQUID-2 has the same fields as SQUID-1.1, although some of the field values have changed.

- **ssl**: This is an ELFF format with custom strings of:

  ```
  date time time-taken c-ip s-action x-rs-certificate-validate-status x-rs-certificate-observed-errors cs-host s-hierarchy s-supplier-name x-rs-connection-negotiated-ssl-version x-rs-connection-negotiated-cipher x-rs-connection-negotiated-cipher-size x-rs-certificate-hostname x-rs-certificate-hostname-category x-cs-connection-negotiated-ssl-version x-cs-connection-negotiated-cipher x-cs-connection-negotiated-cipher-size x-cs-certificate-subject s-ip s-sitename
  ```

- **streaming**: This is an ELFF format with custom strings of:

  ```
  c-ip date time c-dns cs-uri-scheme cs-host cs-uri-port cs-uri-path cs-uri-query c-starttime x-duration c-rate c-status c-playerid c-playerversion c-playerlanguage cs(User-Agent) cs(Referer) c-hostexe c-hostexever c-os c-osversion c-cpu filelength filesize avgbandwidth protocol transport audiocodec videocodec channelURL sc-bytes c-bytes s-pkts-sent c-pkts-received c-pkts-lost-client c-pkts-lost-net c-pkts-lost-cont-net c-resendreqs c-pkts-recovered-ECC c-pkts-recovered-resent c-buffercount c-totalbuffertime c-quality s-ip s-dns s-totalclients s-cpu-util x-cache-user x-cache-info x-client-address
  ```

- **surfcontrol**, **surfcontrolv5**, and **smartfilter**: These are reserved formats that cannot be edited.

- **websense**: This is a reserved format that cannot be edited.

- **bcreportermain_v1**: This is a reserved format that cannot be edited:

  ```
  date time time-taken c-ip cs-username cs-auth-group x-exception-id sc-filter-result cs-categories cs(Referer) sc-status s-action cs-method rs(Content-Type) cs-uri-scheme cs-host cs-uri-port cs-uri-path cs-uri-query cs-uri-extension cs(User-Agent) s-ip sc-bytes cs-bytes x-virus-id
  ```

- **bcreporterssl_v1**: This is a reserved format that cannot be edited. It only contains fields that do not reveal private or sensitive information, unlike the **bcreportermain_v1** format:

  ```
  date time time-taken c-ip cs-username cs-auth-group x-exception-id sc-
  filter-result cs-categories sc-status s-action cs-method rs(Content-
  Type) cs-uri-scheme cs-host cs-uri-port cs-uri-extension cs(User-
  Agent) s-ip sc-bytes cs-bytes x-virus-id x-rs-certificate-observed-
  errors x-rs-connection-negotiated-cipher-strength x-rs-certificate-
  hostname x-rs-certificate-hostname-category
  ```

- **bcreportercifs_v1**: This is a reserved format that cannot be edited:

  ```
  date time c-ip c-port r-ip r-port x-cifs-uid x-cifs-tid x-cifs-fid x-
  cifs-method x-cifs-server x-cifs-share x-cifs-path x-cifs-orig-path x-
  cifs-client-bytes-read x-cifs-server-bytes-read x-cifs-bytes-written
  x-client-connection-bytes x-server-connection-bytes x-server-adn-
  connection-bytes x-cifs-client-read-operations x-cifs-client-write-
  operations x-cifs-client-other-operations x-cifs-server-operations s-
  action x-cifs-error-code cs-username cs-auth-group s-ip
  ```

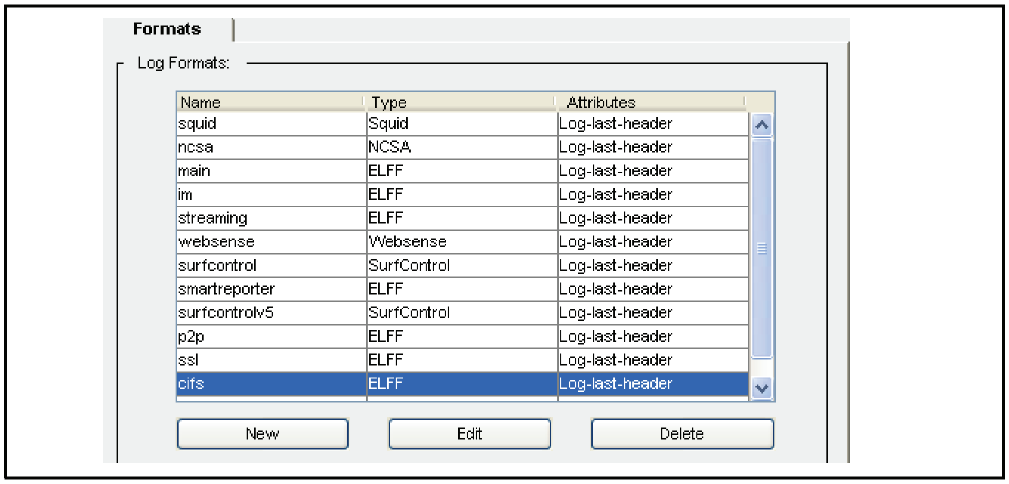
**Note:** If you had previously created formats with the name **smartreporter** or **surfcontrolv5** and you upgrade the device, those formats are changed to **smartreporter_user** or **surfcontrolv5_user**. If you already have a log format named **smartreporter_user** or **surfcontrolv5_user**, then the names become **smartreporter_user1** or **surfcontrolv5_user1**. This naming protocol continues (**_user2**, **_user3**...) as necessary. The logs associated with these formats are automatically associated with the new format name.

## Creating a Custom or ELFF Log Format

Complete the following steps to create a custom or ELFF log format.

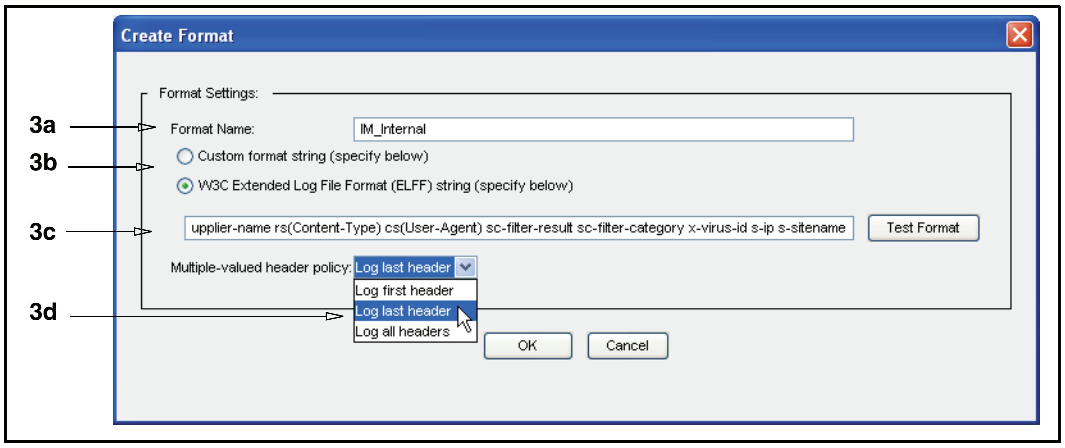
**To create or edit the log format:**

1. Select **Configuration** > **Access Logging** > **Formats**.



| Name | Type | Attributes |
|---|---|---|
| squid | Squid | Log-last-header |
| ncsa | NCSA | Log-last-header |
| main | ELFF | Log-last-header |
| im | ELFF | Log-last-header |
| streaming | ELFF | Log-last-header |
| websense | Websense | Log-last-header |
| surfcontrol | SurfControl | Log-last-header |
| smartreporter | ELFF | Log-last-header |
| surfcontrolv5 | SurfControl | Log-last-header |
| p2p | ELFF | Log-last-header |
| ssl | ELFF | Log-last-header |
| cifs | ELFF | Log-last-header |

2. Click **New** (or highlight a format and click **Edit**). The Create Format dialog displays. If you select an unconfigurable format, you receive an error message.



Format Name: IM_Internal

( ) Custom format string (specify below)

(●) W3C Extended Log File Format (ELFF) string (specify below)

upplier-name rs(Content-Type) cs(User-Agent) sc-filter-result sc-filter-category x-virus-id s-ip s-sitename

Multiple-valued header policy: Log last header

3. Create or modify the format:
   a. Give the format a meaningful name.
   b. Select **Custom format string** (to manually add your own format field)s or **W3C ELFF** (to customize using the standard format fields).
   c. Add log formats or remove from the current list.

   **Note:** ELFF strings cannot start with spaces.

   The access log ignores any ELFF or custom format fields it does not understand. In a downgrade, the format still contains all the fields used in the upgraded version, but only the valid fields for the downgraded version display any information.

   d. Click **Test Format** to test whether the format-string syntax is correct. A line displays below the field that indicates that testing is in progress and then gives a result, such as **Format is valid**.

   **Note:** To doublecheck the format-string syntax, see "Creating a Custom or ELFF Log Format" on page 11.

   e. From the **Multiple-valued header policy** drop-down list, select a header to log: **Log last header**, **log first header**, **log all headers**. This allows you to determine what happens with HTTP-headers that have multiple headers.
   f. Click **OK**.
4. Select **Apply** to commit the changes to the SG appliance.

### *Related CLI Syntax to Manage Access Logging*

❐ To enter configuration mode:

```
SGOS#(config) access-log
```

The following subcommands are available:

```
SGOS#(config access-log) create log log_name
SGOS#(config access-log) create format format_name
SGOS#(config access-log) cancel-upload all
SGOS#(config access-log) cancel-upload log log_name
```

SGOS#(config access-log) **default-logging {cifs | epmapper | ftp | http | https-forward-proxy | https-reverse-proxy | icp | im | mapi | mms | p2p | rtsp | socks | ssl | tcp-tunnel | telnet}** *log_name*

SGOS#(config access-log) **delete log** *log_name*

SGOS#(config access-log) **delete format** *format_name*

SGOS#(config access-log) **disable**

SGOS#(config access-log) **early-upload** ***megabytes***

SGOS#(config access-log) **edit log** *log_name*—changes the prompt to SGOS#(config edit log *log_name*)

SGOS#(config access-log) **edit format** *format_name*—changes the prompt to SGOS#(config edit format *format_name*)

SGOS#(config access-log) **enable**

SGOS#(config access-log) **exit**

SGOS#(config access-log) **max-log-size** *megabytes*

SGOS#(config access-log) **no default-logging {cifs | epmapper | ftp | http | https-forward-proxy | https-reverse-proxy | icp | im | mapi | mms | p2p | rtsp | socks | ssl | tcp-tunnel | telnet}**

SGOS#(config access-log) **overflow-policy delete**

SGOS#(config access-log) **overflow-policy stop**

SGOS#(config access-log) **upload all**

SGOS#(config access-log) **upload log** *log_name*

SGOS#(config access-log) **view**

SGOS#(config access-log) **view [log [brief |** *log_name***]]**

SGOS#(config access-log) **view [format [brief |** *format_name***]]**

SGOS#(config access-log) **view [statistics [***log_name***]]**

SGOS#(config access-log) **view [default-logging]**

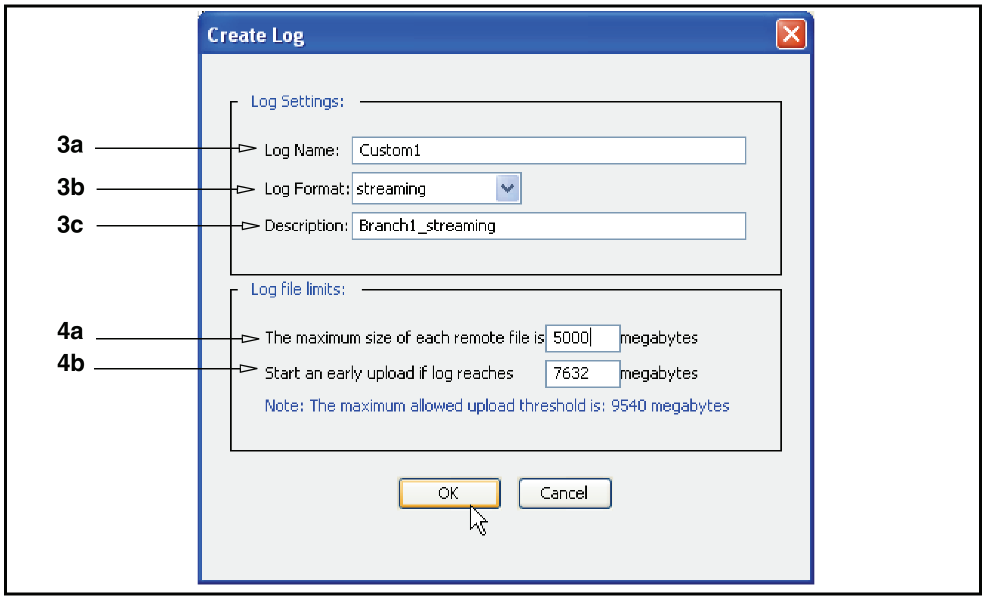
# Chapter 3: Creating and Editing Access Log Facility

You can use existing log facilities and modify them for your needs. You can also create new log facilities for special circumstances, such as associating the SurfControl log format with a log facility. To create new log facilities, continue with the next section. To edit an existing log facility, skip to "Configuring Global Settings" on page 18.

**Note:** Several log facilities have already been created. Before creating a new one, check the existing ones to see if they fit your needs. If you want to use a custom log format with the new log facility, you must create the log format before associating it with a log (see Chapter 2: "Creating and Editing Log Formats" on page 9).

**To create a log facility:**

1. Select **Configuration > Access Logging > Logs > Logs**.
2. The log facilities already created are displayed in the **Logs** tab. To create a new log, click **New**.



Log Settings:

Log Name: Custom1

Log Format: streaming

Description: Branch1_streaming

Log file limits:

The maximum size of each remote file is 5000 megabytes

Start an early upload if log reaches 7632 megabytes

Note: The maximum allowed upload threshold is: 9540 megabytes

3. Fill in the fields as appropriate:
   a. **Log Name**: Enter a log facility name that is meaningful to you.
   b. **Log Format**: Select a log format from the drop-down list.
   c. **Description:** Enter a meaningful description of the log. It is used for display purposes only.
4. Fill in the **Log file limits** panel as appropriate. (You can edit these settings later. See "Configuring Global Settings" on page 18.)

a. The maximum size for each remote log file (the file on the upload server) defaults to **0**, meaning that all data is sent to the same log file. If you set a maximum size, a new log file opens when the file reaches that size. This setting is valid for both periodic and continuous uploads.

b. Specify a size that triggers an early upload—the maximum upload size varies depending on the size of the appliance disks (the maximum allowed upload threshold appears below this field).

5. Click **OK**.

6. Select **Apply** to commit the changes to the SG appliance.

## Editing an Existing Log Facility

Several facilities exist, each associated with a log format. For a description of the format, see "Chapter 3: Creating and Editing Access Log Facility" .

- ❐ **im** (Instant Messaging): Associated with the im format.
- ❐ **main**: Associated with the main format.
- ❐ **p2p** (Peer-to-Peer): Associated with the p2p format.
- ❐ **ssl:** Associated with the SSL format.
- ❐ **streaming**: Associated with the streaming format.

Use the following procedures to edit log facilities you have created.

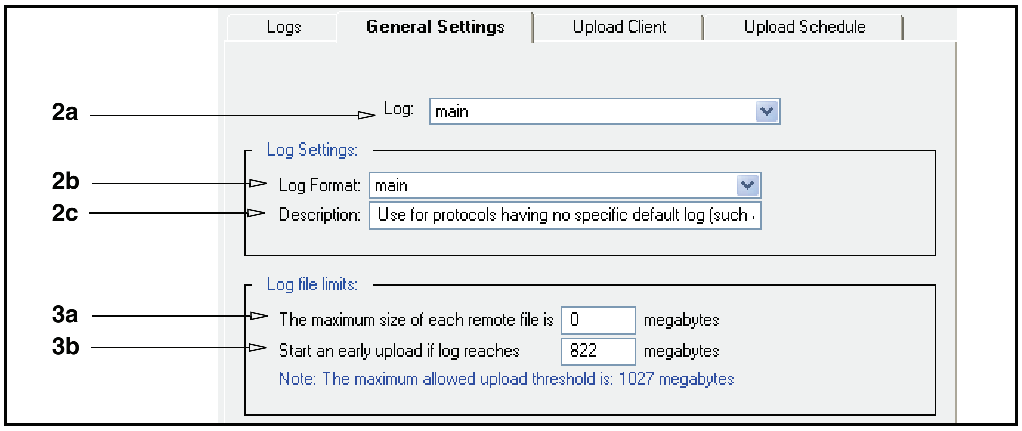
**Note:** If you change the log format of a log, remember that ELFF formats require an ELFF header in the log (the list of fields being logged are mentioned in the header) and that non-ELFF formats do not require this header.

The format of data written to the log changes as soon as the format change is applied; for best practices, do a log upload before the format change and immediately after (to minimize the number of log lines in a file with mixed log formats).

Upload the log facility before you switch the format.

**To edit an existing log facility:**

1. Select **Configuration > Access Logging > Logs > General Settings**.



Log: main

Log Format: main

Description: Use for protocols having no specific default log (such

The maximum size of each remote file is 0 megabytes

Start an early upload if log reaches 822 megabytes

Note: The maximum allowed upload threshold is: 1027 megabytes

2. Fill in the fields as appropriate:
   a. **Log**: Select an already-existing log facility from the **Log** drop-down list.
   b. **Log Format**: Select the log format from the drop-down list.
   c. **Description:** Enter a meaningful description of the log. (If you chose an existing log format, the default description for that log is displayed. You can change it.)
3. Fill in the **Log file limits** panel as appropriate:
   a. The maximum size for each remote log file (the file on the upload server) defaults to **0**, meaning that all data is sent to the same log file. If you set a maximum size, a new log file opens when the file reaches that size. This setting is valid for both periodic and continuous uploads.
   b. Specify a size that triggers an early upload—the maximum upload size varies depending on the size of the appliance disks (the maximum allowed upload threshold appears below this field).
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

## Associating a Log Facility with a Protocol

You can associate a log facility with a protocol at any point in the process. By default, new systems have specific protocols associated with specific logs. This allows you to begin access logging as soon as it is enabled (see Chapter 3: "Creating and Editing Access Log Facility" on page 15).

**Note:** If you have a policy that defines protocol and log association, that policy overrides any settings you make here.

The following list shows the protocols supported and the default log facilities assigned to them, if any:

Table 3-1. Default Log Facility Assignments

| Protocol | Assigned Default Log Facility |
|---|---|
| Endpoint Mapper | main |
| FTP | main |
| HTTP | main |
| HTTPS-Reverse-Proxy | main (Set to the same log facility that HTTP is using upon upgrade.) |
| HTTPS-Forward-Proxy | ssl (If the facility for HTTP, TCP, or SOCKS is set before upgrade.) |
| ICP | none |
| Instant Messaging | im |
| MAPI | mapi |
| Peer to Peer | p2p |

Table 3-1. Default Log Facility Assignments (Continued)

| Protocol | Assigned Default Log Facility |
|---|---|
| RealMedia/QuickTime | streaming |
| SOCKS | none |
| SSL | ssl (If the facility for HTTP, TCP or SOCKS is set before upgrade.) |
| TCP Tunnel | main |
| Telnet | main |
| Windows Media | streaming |

**Note:** To disable access logging for a particular protocol, you must either disable the default logging policy for that protocol (see "Disabling Access Logging for a Particular Protocol" on page 18) or modify the access logging policy in VPM (refer to *Volume 7: VPM and Advanced Policy*).

**To associate a log facility with a protocol:**

1. Select **Configuration > Access Logging > General > Default Logging**.
2. Highlight the protocol you want to associate with a log facility and click **Edit**.
3. Select a log facility from the **Default Log** drop-down list.

   **Note:** To disable access logging for that protocol, select **none**.

4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

## Disabling Access Logging for a Particular Protocol

**To disable access logging for a particular protocol:**

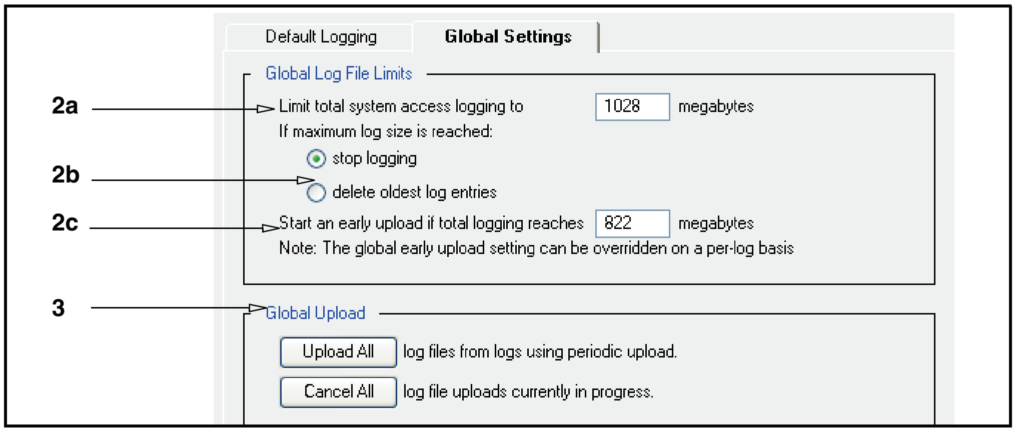
1. Select **Configuration > Access Logging > General > Default Logging**.
2. Highlight the protocol to disable access logging and click **Edit**.
3. Select **none** from the drop-down menu.
4. Click **OK**.
5. Select **Apply** to commit the changes to the SG appliance.

## Configuring Global Settings

You can set global limits for log size and early upload times. These settings can be overridden by individual log facilities.

**To set global log facility limits:**

1. Select **Configuration > Access Logging > General > Global Settings**.



Limit total system access logging to 1028 megabytes

If maximum log size is reached:

[✓ stop logging] delete oldest log entries

Start an early upload if total logging reaches 822 megabytes

Note: The global early upload setting can be overridden on a per-log basis

Upload All log files from logs using periodic upload.

Cancel All log file uploads currently in progress.

2. Fill in the **Global Log File Limits** panel as appropriate:
   a. Configure the maximum size occupied by all of the log files (in megabytes).
   b. Determine the behavior of the log when the maximum size is reached. You can have the log stop logging (and do an immediate upload) or have it delete the oldest log entries.
   c. Specify the size of the log that triggers an early upload.
3. The **Global Upload** options affect all log facilities currently available. They do not affect scheduled upload times. You can upload logs now, using the periodic upload method, or you can cancel all the uploads that are currently in progress.
4. Select **Apply** to commit the changes to the SG appliance.

# Chapter 4:  Configuring the Upload Client

Blue Coat supports four types of upload client:

- FTP client, the default
- HTTP client
- Custom client
- Websense client

Blue Coat also supports secure FTP, HTTP, and Custom client.

The Custom client can be used for special circumstances, such as working with SurfControl Reporter. Custom client is based on plain sockets.

**Note:** You must have a socket server to use the Custom client.

The general options you enter in the **Upload Client** tab affect all clients. Specific options that affect individual clients are discussed in the FTP client, HTTP client, Custom client, or Websense client panes or the `access-log ftp-client`, `https-client`, `custom-client`, or `websense-client` CLI commands.

Only one client can be used at any one time. All four can be configured, but only the selected client is used.

The SGOS software provides access logging with two types of uploads to a remote server:

- continuous uploading, where the device continuously streams new access log entries from the device memory to a remote server.
- scheduled (periodic) uploading, where the device transmits log entries on a scheduled basis. See Chapter 5: "Configuring the Upload Schedule" for more information.

The SGOS software allows you to upload either compressed access logs or plain-text access logs. The device uses the gzip format to compress access logs. Gzip-compressed files allow more log entries to be stored in the device. Advantages of using file compression include:

- Reduces the time and resources used to produce a log file because fewer disk writes are required for each megabyte of log-entry text.
- Uses less bandwidth when the device sends access logs to an upload server.
- Requires less disk space.

Compressed log files have the extension `.log.gz`. Text log files have the extension `.log`.

**Note:** You cannot upload gzip access-log files for the Websense client.

For greater security, you can configure the SGOS software to

- encrypt the access log

- sign the access log

## Encrypting the Access Log

To encrypt access log files, you must first place an external certificate on the SG appliance (see "Importing an External Certificate" on page 22). The device derives a session key from the public key in the external certificate and uses it to encrypt the log. When an access log is encrypted, two access log files are produced: an ENC file (extension `.enc`), which is the encrypted access log file, and a DER file (extension `.der`), which contains the SG appliance session key and other information. You need four things to decrypt an encrypted access log:

- The ENC file
- The DER file
- The external (public key) certificate
- The corresponding private key

For information about decrypting a log, see "Decrypting an Encrypted Access Log" on page 26.

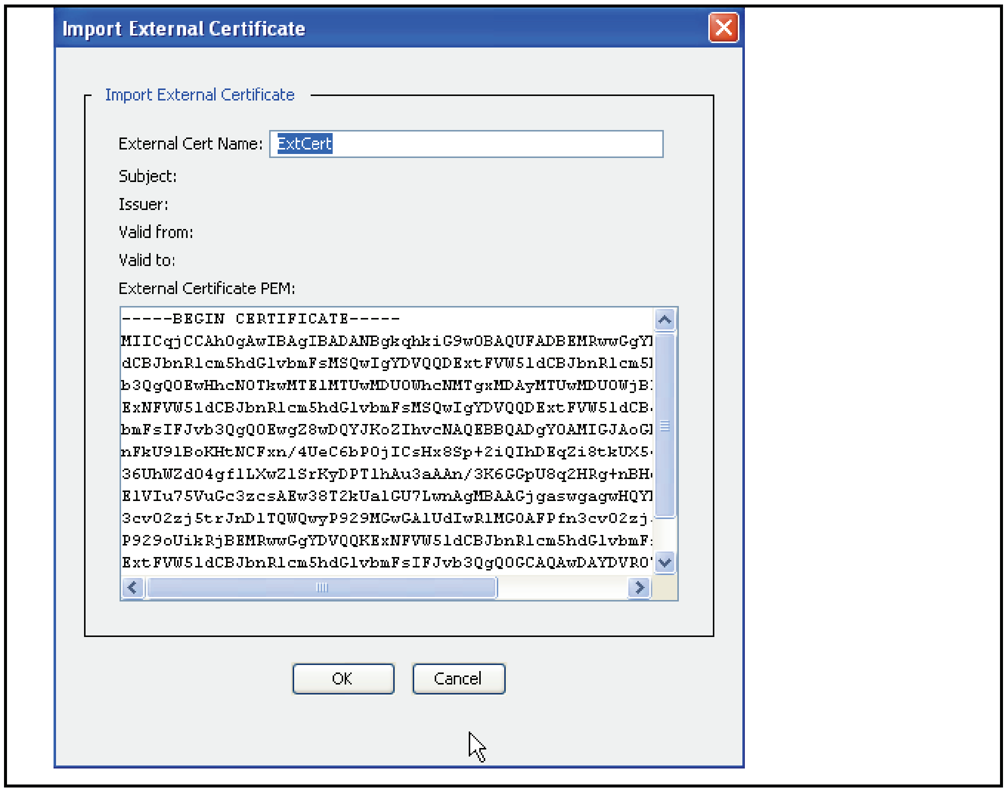
**Note:** The encryption feature is not available for custom or Websense clients.

## Importing an External Certificate

You can import an X.509 certificate into the SG appliance to use for encrypting data.

**To Import an external certificate:**

1. Copy the certificate onto the clipboard.
2. Select **Configuration > SSL > External Certificates**.
3. Click **Import**.



External Cert Name: ExtCert
Subject:
Issuer:
Valid from:
Valid to:
External Certificate PEM:

```
-----BEGIN CERTIFICATE-----
MIICqjCCAh0gAwIBAgIBADANBgkqhkiG9w0BAQUFADBEMRwwGgY
dCBJbnRlcm5hdGlvbmFsMSQwIgYDVQQDExtFVW5ldCBJbnRlcm5
b3QgQ0EwHhcNOTkwMTE1MTUwMDU0WhcNMTgxMDAyMTUwMDU0WjB
ExNFVW5ldCBJbnRlcm5hdGlvbmFsMSQwIgYDVQQDExtFVW5ldCB
bmFsIFJvb3QgQ0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoG
nFkU91BoKHtNCFxn/4UeC6bP0jICsHx8Sp+2iQIhDEqZi8tkUX5
36UhWZd04gflLXwZlSrKyDPTlhAu3aAAn/3K6GGpU8q2HRg+nBH
ElVIu75VuGc3zcsAEw38T2kUalGU7LwnAgMBAAGjgaswgagwHQY
3cv02zj5trJnDlTQWQwyP929MGwGA1UdIwRlMG0AFPfn3cv02zj
P929oUikRjBEMRwwGgYDVQQKExNFVW5ldCBJbnRlcm5hdGlvbmF
ExtFVW5ldCBJbnRlcm5hdGlvbmFsIFJvb3QgQ0GCAQAwDAYDVR0
```

4. Enter the name of the external certificate into the **External Cert Name** field and paste the certificate into the **External Certificate** field. Be sure to include the `----BEGIN CERTIFICATE----` and `-----END CERTIFICATE----` statements.

5. Click **OK**.

6. Select **Apply** to commit the changes to the SG appliance.

### *Deleting an External Certificate*

**To delete an external certificate:**

1. Select **Configuration > SSL > External Certificates**.

2. Highlight the name of the external certificate to be deleted.

3. Click **Delete**.

4. Click **OK** in the Confirm delete dialog that appears.

5. Select **Apply** to commit the changes to the SG appliance.

## Digitally Signing Access Logs

You can digitally sign access logs to certify that a particular SG appliance wrote and uploaded this log file. Signing is supported for both content types— text and gzip—and for both upload types—continuous and periodic. Each log file has a signature file associated with it that contains the certificate and the digital signature for verifying the log file. The signature file has the same name as the access log file but with a .sig extension; that is, `*filename*.log.sig`, if the access log is a text file, or `*filename*.log.gzip.sig`, if the access log is a gzip file.

**Note:** Signing is disabled by default.

You can digitally sign your access log files with or without encryption. If the log is both signed and encrypted, the signing operation is done first, meaning that the signature is calculated on the unencrypted version of the file. You must decrypt the log file before verifying the file. Attempting to verify an encrypted file fails.

When you create a signing keyring (which must be done before you enable digital signing), keep in mind the following:

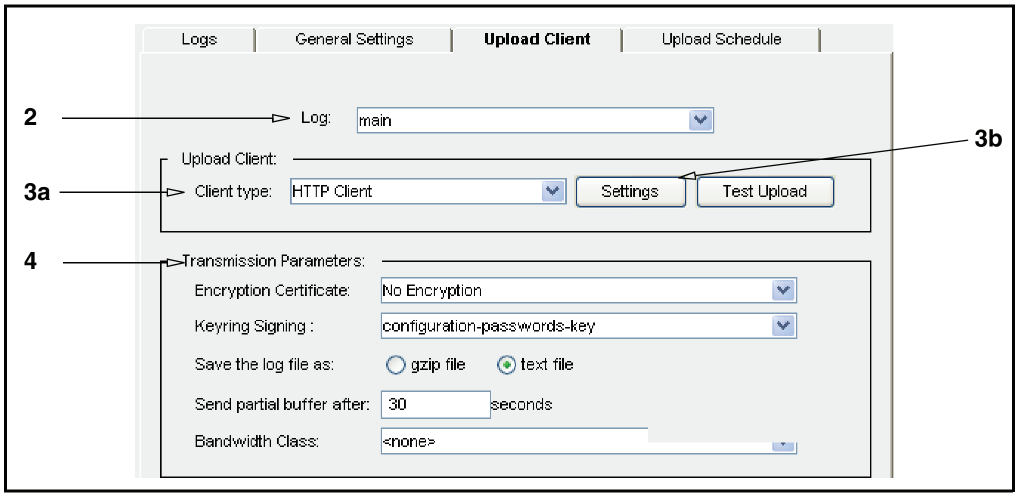
- The keyring must include an external certificate. (An external certificate is one for which the SG appliance does not have the private key.).
- The certificate purpose must be set for **smime** signing**.** If the certificate purpose is set to anything else, you cannot use the certificate for signing.
- Add the `%c` parameter in the filenames format string to identify the keyring used for signing. If encryption is enabled along with signing, the `%c` parameter expands to *`keyringName_Certname`*.

**Note:** The signing feature is not available for custom or Websense clients.

For information about verifying a log, see "Verifying a Digital Signature" on page 26.

**To configure the upload client:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.



Log: main

Upload Client:

Client type: HTTP Client

Transmission Parameters:

Encryption Certificate: No Encryption

Keyring Signing : configuration-passwords-key

Save the log file as: gzip file [✓ text file]

Send partial buffer after: 30 seconds

Bandwidth Class: <none>

2. From the **Log** drop-down list, select the log facility to configure. The facility must exist before it displays in this list.
3. Select and configure the client type:
   a. From the **Client type** drop-down list, select the upload client to use. Only one client can be configured for each log facility.
   b. Click **Settings** to customize the upload client.

   For information on customizing the clients, skip to "Editing the FTP Client" on page 27, "Editing the HTTP Client" on page 28, "Editing the Custom Client" on page 29, "Editing the Custom SurfControl Client" on page 30, or "Editing the Websense Client" on page 31.

For information about testing the upload client, see Chapter 4: "Configuring the Upload Client".

4. Configure **Transmission Parameters**, if applicable:
   a. (Optional) To use an external certificate to encrypt the uploaded log facility, select an external certificate from the **Encryption Certificate** drop-down list. You must first import the external certificate to the SG appliance (see "Importing an External Certificate" on page 22).

      The encryption option is not available for Websense or Custom clients.
   b. (Optional) To enable the digital signature of the uploaded access log, select a keyring from the **Keyring Signing** drop-down list. The signing keyring, with a certificate set to **smime**, must already exist. A certificate set to any other purpose cannot be used for digital signatures.

      The digital signing option is not available for Websense or Custom clients.
   c. Select one of the **Save the log file as** radio buttons to determine whether the access log that is uploaded is compressed (**gzip file**, the default) or not (**text file**).

      **Note:** If you are configuring a SurfControl Custom client, select the **text file** radio button.

      If you chose **text file**, you can change the **Send partial buffer after** ***n*** **seconds** field to the time you need (30 seconds is the default).

      This field configures the maximum time between text log packets, meaning that it forces a text upload after the specified length of time even if the internal log buffer is not full. If the buffer fills up before the time specified in this setting, the text uploads right away, and is not affected by this maximum setting.

      **Note:** If you chose **gzip file**, the **Send partial buffer after** ***n*** **seconds** field is not configurable. Also, this setting is only valid for continuous uploading (see Chapter 5: "Configuring the Upload Schedule" for information about continuous uploading).
   d. (Optional) To manage the bandwidth for this log facility, select a bandwidth class from the **Bandwidth Class** drop-down list.

      The default setting is **none**, which means that bandwidth management is disabled for this log facility by default.

   **Note:** Before you can manage the bandwidth for this log facility, you must first create a bandwidth-management class. It is the log facility that is bandwidth-managed—the upload client type does not affect this setting. Refer to *Volume 6: Advanced Networking* for information about enabling bandwidth management and creating and configuring the bandwidth class.

   Less bandwidth slows down the upload, while more could flood the network.

5. Select **Apply** to commit the changes to the SG appliance.

## Disabling Log Uploads

To disable log uploads, set the upload client-type to none.

**To disable an upload:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.
2. Select the log facility for which you want to disable an upload from the **Log** drop-down menu.
3. Select **NONE** from the **Client type** drop-down menu.
4. Select **Apply** to commit the changes to the SG appliance.

## Decrypting an Encrypted Access Log

To decrypt an encrypted access log, you must concatenate the DER and ENC files (with the DER file in front of the ENC file) and use a program such as OpenSSL for decryption. For example, use the following UNIX command and a tool such as OpenSSL to concatenate the DER and ENC files and decrypt the resulting file:

```
cat path/filename_of_DER_file path/filename_of_ENC_file | openssl
smime -decrypt -inform DER -binary -inkey path/filename_of_private_key
-recip path/filename_of_external_certificate -out path/
filename_for_decrypted_log_file
```

You can also download a script based on the OpenSSL tool for decryption. Go to https://download.bluecoat.com/release/SG4/files/accesslog_decrypt.zip.

## Verifying a Digital Signature

If the file whose digital signature you want to verify is also encrypted, you must decrypt the file prior to verifying the signature. (See "Decrypting an Encrypted Access Log" on page 26 above for more information.)

You can use a program such as OpenSSL to verify the signature. For example, use the following command in OpenSSL:

```
openssl smime -CAfile cacrt -verify -in filename.sig -content
filename.log -inform DER -out logFile
```

where

| | |
|---|---|
| *cacrt* | The CA certificate used to issue the certificate in the signature file. |
| *filename.sig* | The file containing the digital signature of the log file. |
| *filename.log* | The log file generated after decryption. If the access log is a gzip file, it contains a `.gz` extension. |
| *logFile* | The filename that is generated after signature verification. |

## Editing Upload Clients

Four upload clients are supported by Blue Coat: FTP, HTTP, Custom, and Websense. Each of these clients are described below. You can also create a SurfControl or SmartFilter upload client.

Multiple upload clients can be configured per log facility, but only one can be enabled and used per upload.

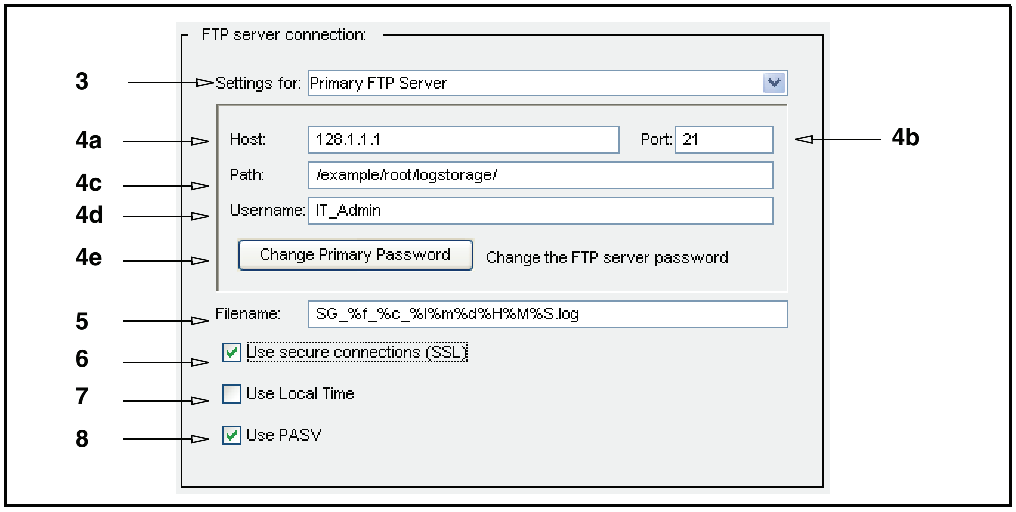
## Editing the FTP Client

**To edit the FTP client:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.

   See Chapter 4: "Configuring the Upload Client" for configuration information.

2. Select **FTP Client** from the **Client type** drop-down list. Click the **Settings** button.



FTP server connection:

Settings for: Primary FTP Server

Host: 128.1.1.1 Port: 21

Path: /example/root/logstorage/

Username: IT_Admin

Change Primary Password — Change the FTP server password

Filename: SG_%f_%c_%l%m%d%H%M%S.log

[✓] Use secure connections (SSL)

[ ] Use Local Time

[✓] Use PASV

3. Select the primary or alternate FTP server to configure from the **Settings for** drop-down list.

4. Fill in the server fields, as appropriate:

   a. **Host**: The name of the upload client host. If the **Use secure connections (SSL)** checkbox is selected, the hostname must match the hostname in the certificate presented by the server.

   b. **Port**: The default is 21; it can be changed.

   c. **Path**: The directory path where the access log is uploaded on the server.

   d. **Username**: This is the username that is known on the host you are configuring.

   e. **Change Password**: Change the password on the FTP; the Change Password dialog displays; enter and confirm the new password; click **OK**.

5. **Filename**: The **Filename** field is comprised of text and/or specifiers. The default filename includes specifiers and text that indicate the log name (`%f`), name of the external certificate used for encryption, if any (`%c`), the fourth parameter of the SG appliance IP address (`%l`), the date and time (Month: `%m`, Day: `%d`, Hour: `%H`, Minute: `%M`, Second: `%S`), and the `.log` or `.gzip.log` file extension.

**Note:** Be cautious if you change the **Filename** field. If an ongoing series of access logs files are produced and you do not have time-specifiers in this field, each access log file produced overwrites the old file. Also, if you use more than one external certificate to encrypt logs, include the `%c` specifier in the **Filename** field to keep track of which external certificate was used to encrypt the uploaded log file.

If you are creating a SurfControl client, change the `.log` file extension to `.tmp`.

6. **Secure Connections**: If you use FTPS, select the **Use secure connections (SSL)** checkbox. The remote FTP server must support FTPS.
7. **Local Time**: If you want the upload to reflect the local time it was uploaded instead of Universal Time Coordinates (UTC), select **Local Time**.
8. **Use PASV**: With **Use PASV** selected (the default), the SG appliance connects to the FTP server. With **Use PASV** de-selected, the FTP server uses the PORT command to connect to the SG appliance.
9. Click **OK**.
10. Select **Apply** to commit the changes to the SG appliance.

## Editing the HTTP Client

Access log uploads done through an HTTP/HTTPS client use the HTTP PUT method. The destination HTTP server (where the access logs are being uploaded) must support this method. Microsoft's IIS allows the server to be directly configured for write (PUT/ DELETE) access. Other servers, such as Apache, require installing a new module for the PUT method for access log client uploads.

You can create either an HTTP or an HTTPS upload client through the HTTP Client dialog. (Create an HTTPS client by selecting **Use secure connections (SSL)**.)

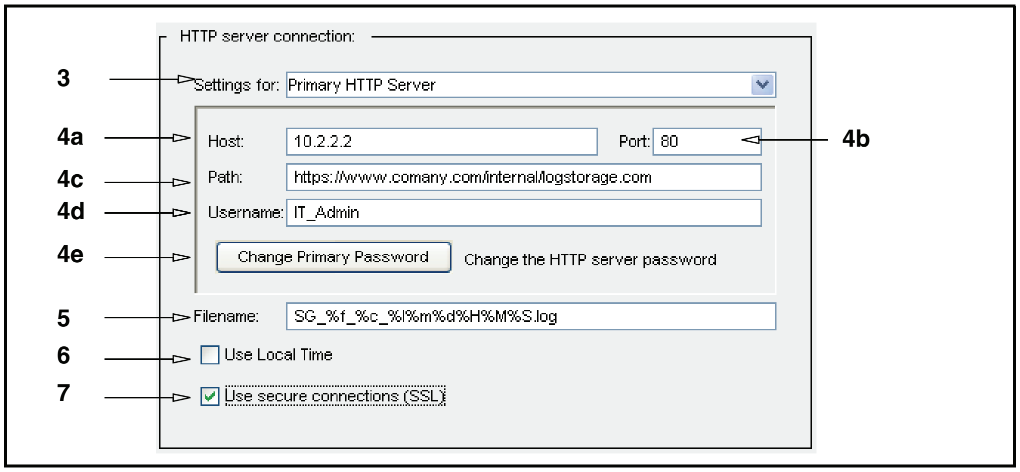
**Note:** To create an HTTPS client, you must also import the appropriate CA Certificate. For information, refer to *Volume 3: Proxies and Proxy Services*.

**To edit the HTTP client:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.

   See Chapter 4: "Configuring the Upload Client" on page 21 for configuration information.

2. Select **HTTP Client** from the **Client type** drop-down list. Click **Settings**.



Settings for: Primary HTTP Server
Host: 10.2.2.2
Port: 80
Path: https://www.comany.com/internal/logstorage.com
Username: IT_Admin
Filename: SG_%f_%c_%l%m%d%H%M%S.log
Use Local Time: not selected
Use secure connections (SSL): selected

3. From the **Settings for** drop-down list, select the primary or alternate HTTP server to configure.

4. Fill in the server fields, as appropriate:
   a. **Host**: The name of the upload host. If **Use secure connections (SSL)** is selected, the hostname must match the hostname in the certificate presented by the server.
   b. **Port**: The default is 80, but you can change it.

   **Note:** For HTTPS, change the port to 443.

   c. **Path**: The directory path where the access log facility is uploaded on the server.
   d. **Username**: This is the username that is known on the host you are configuring.
   e. **Change Password**: Change the password on the HTTP host; the Change Password dialog displays; enter and confirm the new password and click **OK**.
5. **Filename**: The **Filename** field is comprised of text and/or specifiers. The default filename includes specifiers and text that indicate the log name (`%f`), name of the external certificate used for encryption, if any (`%c`), the fourth parameter of the SG appliance IP address (`%l`), the date and time (Month: `%m`, Day: `%d`, Hour: `%H`, Minute: `%M`, Second: `%S`), and the `.log` or `.gzip.log` file extension.

   **Note:** Be cautious if you change the **Filename** field. If an ongoing series of access log files are produced and you do not have time-specifiers in this field, each access log file produced overwrites the old file. Also, if you use more than one external certificate to encrypt logs, include the `%c` specifier in the **Filename** field to keep track of which external certificate can decrypt the uploaded log file.

   If you are creating a SurfControl client, change the `.log` file extension to `.tmp`.

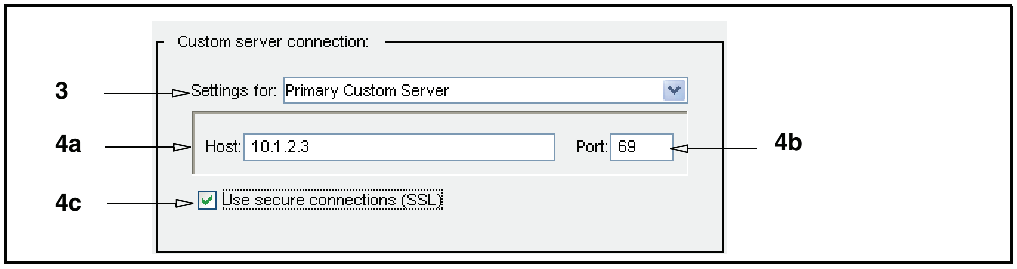
6. **Local Time**: If you want the upload to reflect the local time it was uploaded instead of Universal Time Coordinate (UTC), select **Local Time**.
7. **Use secure connections (SSL)**: Select this to create an HTTPS client. To create an HTTPS client, you must also create a keypair, import or create a certificate, and, if necessary, associate the keypair and certificate (called a keyring), with the SSL-client.
8. Click **OK**.
9. Select **Apply** to commit the changes to the SG appliance.

## Editing the Custom Client

**To edit the custom client:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.

   See Chapter 4: "Configuring the Upload Client" for configuration information.
2. Select **Custom Client** from the **Client type** drop-down list. Click the **Settings** button.



Settings for: Primary Custom Server
Host: 10.1.2.3
Port: 69
☑ Use secure connections (SSL)

3. From the **Settings for** drop-down list, select to configure the primary or alternate custom server.
4. Fill in the server fields, as appropriate:
   a. **Host**: Enter the hostname of the upload destination. If **Use secure connections (SSL)** is selected, the hostname must match the hostname in the certificate presented by the server.
   b. **Port**: The default is 69; it can be changed.
   c. **Use secure connections (SSL)**: Select this if you are using secure connections.
5. Click **OK**.
6. Select **Apply** to commit the changes to the SG appliance.

## Editing the Custom SurfControl Client

You can use the Custom Client to create an upload client that uploads information to SurfControl Reporter. Before you begin, verify that:

❒ You have created a log (see Chapter 3: "Creating and Editing Access Log Facility").

❒ You have associated the SurfControl log format with the log you created (see Chapter 3: "Creating and Editing Access Log Facility").

**To edit the SurfControl client:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.

   See Chapter 4: "Configuring the Upload Client" for configuration information.
2. From the **Log** drop-down list, select the SurfControl log that you associated with the SurfControl log format.
3. Verify the **Save the log file as** radio button is set to **text file**, not **gzip file**.
4. Select **Custom Client** from the **Client type** drop-down list.

   **Note:** For specific information on managing upload clients, see "Editing the Custom Client" on page 29.
5. Click the **Settings** button for that client.
6. Customize the upload client for SurfControl Reporter.

a. Enter the hostname, path, and username, if necessary, for the SurfControl Reporter server.
b. Make sure the filename extension is `.tmp` and not `.gzip` or `.log`. SurfControl only recognizes files with a `.tmp` extension.
c. If your SurfControl server supports SSL, select the **Use secure connections (SSL)** checkbox.

7. Click **OK**.
8. Select **Apply** to commit the changes to the SG appliance.

## Editing the Websense Client

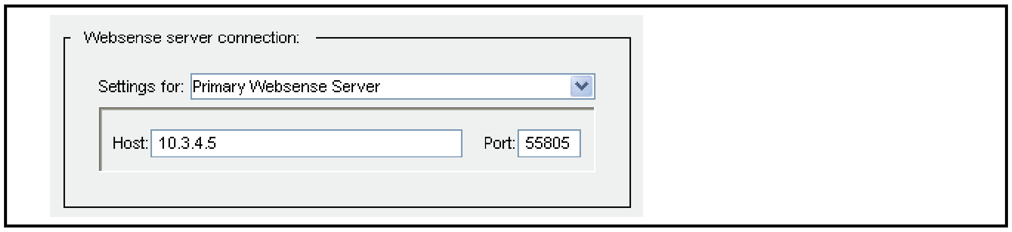
Before you begin, make sure you have created a Websense log using the Websense log format and configured the log to your environment. See Chapter 3: "Creating and Editing Access Log Facility".

**Note:** You cannot upload gzip access log files with the Websense client.

**To edit the Websense client:**

1. Select **Configuration > Access Logging > Logs > Upload Client**.

   See Chapter 4: "Configuring the Upload Client" for configuration information.
2. Select the **Websense Client** from the **Client type** drop-down list. Click **Settings**.


Websense server connection:
Settings for: Primary Websense Server
Host: 10.3.4.5
Port: 55805

3. From the **Settings for** drop-down list, select the primary or alternate server you want to configure.
4. Fill in the fields as appropriate:
   a. **Host**: Enter the hostname of the primary Websense Server.
   b. **Port**: The default is 55805, but you can change it if the Websense Server is using a different port.
5. Repeat for the **Alternate Websense Server**.
6. Click **OK**.
7. Select **Apply** to commit the changes to the SG appliance.

# Chapter 5: Configuring the Upload Schedule

The Upload Schedule allows you to configure the frequency of the access logging upload to a remote server, the time between connection attempts, the time between keep-alive packets, the time at which the access log is uploaded, and the protocol that is used.

You can specify either *periodic uploading* or *continuous uploading*. Both periodic and continuous uploading can send log information from an SG appliance farm to a single log analysis tool. This allows you to treat multiple appliances as a single entity and to review combined information from a single log file or series of related log files.

With periodic uploading, the SGOS software transmits log entries on a scheduled basis (for example, once daily or at specified intervals) as entries are batched, saved to disk, and uploaded to a remote server.

**Note:** When you configure a log for continuous uploading, it continues to upload until you stop it. To stop continuous uploading, switch to periodic uploading temporarily. This is sometimes required for gzip or encrypted files, which must stop uploading before you can view them.

With continuous uploading, the SG appliance continuously *streams* new access log entries from the device memory to a remote server. Here, *streaming* refers to the real-time transmission of access log information. The SGOS software transmits access log entries using the specified client, such as FTP client. A keep-alive is sent to keep the data connection open.

Continuous uploading allows you to view the latest logging information almost immediately, send log information to a log analysis tool for real-time processing and reporting, maintain the SG appliance performance by sending log information to a remote server (avoiding disk writes), and save device disk space by saving log information on the remote server.

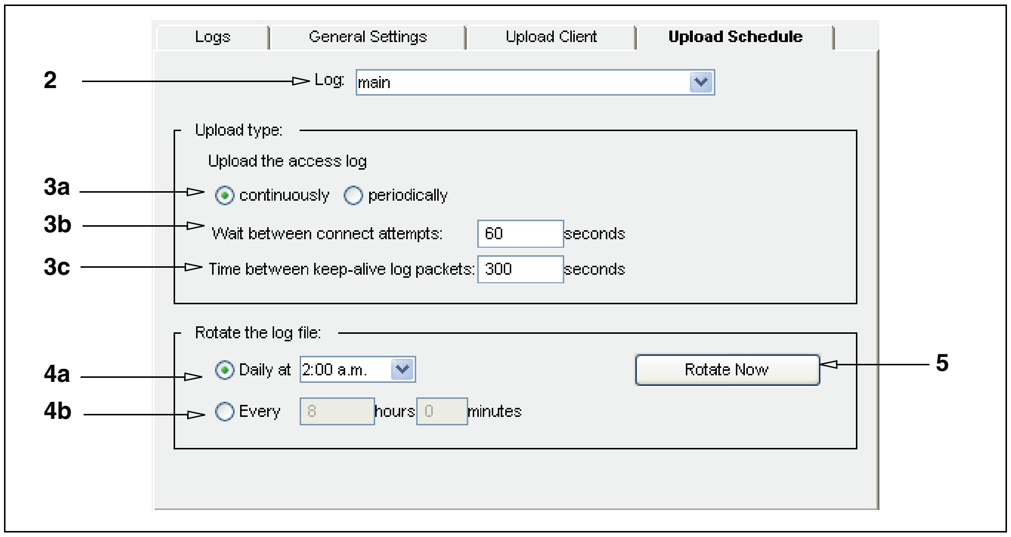
If the remote server is unavailable to receive continuous upload log entries, the SGOS software saves the log information on the device disk. When the remote server is available again, the appliance resumes continuous uploading.

**Note:** If you do not need to analyze the upload entries in real time, use periodic uploading because it is more reliable than continuous uploading.

If there is a problem configuring continuous uploading to Microsoft Internet Information Server (IIS), use periodic uploading instead.

## To configure the upload schedule:

1. Select **Configuration > Access Logging > Logs > Upload Schedule**.



Log: main

Upload type:

Upload the access log

[✓ continuously] periodically

Wait between connect attempts: 60 seconds

Time between keep-alive log packets: 300 seconds

Rotate the log file:

[✓ Daily at] 2:00 a.m.

Every 8 hours 0 minutes

2. From the **Log** drop-down list, select the log type.
3. Select the **Upload Type**:
   a. Select **continuously** (stream access log entries to a remote server) or **periodically** (transmit on a scheduled basis).
   b. To change the time between connection attempts, enter the new time (in seconds) in the **Wait between connect attempts** field.
   c. (Only accessible if you are updating continuously) To change the time between keep-alive packets, enter the new time (in seconds) in the **Time between keep-alive log packets** field.

      Keepalives maintain the connection during low periods of system usage. When no logging information is being uploaded, the SGOS software sends a keep-alive packet to the remote server at the interval you specify, from 1 to 65535 seconds. If you set this to 0 (zero), you effectively disable the connection during low usage periods. The next time that access log information needs to be uploaded, the SG appliance automatically reestablishes the connection.
4. Determine when logs are uploaded or rotated:
   a. (Optional) From the **Daily at** drop-down list, specify the time of day to log update (for periodic uploads) or rotate (for continuous uploads).
   b. (Optional) To have the log uploaded or rotated on a daily basis, select **Every** and enter the time between uploads.
5. **Rotate** or **Upload Now**:
   - Continuous Upload: *Log rotation* helps prevent logs from growing excessively large. Especially with a busy site, logs can grow quickly and become too big for easy analysis. With log rotation, the SGOS software periodically creates a new log file, and archives the older one without disturbing the current log file.
   - Periodic Upload: You can upload the access logs now or you can cancel any access-log upload currently in progress (if you are doing periodic uploads). You can rotate the access logs now (if you are doing continuous uploads). These actions do not affect the next scheduled upload time.

- **Cancel upload** (for periodic uploads) allows you to stop repeated upload attempts if the Web server becomes unreachable while an upload is in progress. Clicking this sets log uploading back to idle if the log is waiting to retry the upload. If the log file is in the process of uploading, it takes time for it to take effect.

6. Select **Apply** to commit the changes to the SG appliance.

## Testing Access Log Uploading

For the duration of the test, configure the event log to use the verbose event level (refer to *Volume 10: Managing the Blue Coat SG Appliance*). This logs more complete log information. After you test uploading, you can check the event log for the test upload event and determine whether any errors occurred (go to **Statistics > Event Logging**). You cannot check the event log.

**To test access log uploading:**

You can do a test access log upload. Before you begin, make sure you have configured the upload client completely.

1. Select **Configuration > Access Logging > Logs > Upload Client**.
2. Click **Test Upload**.
3. Click **OK** in the Test upload dialog.
4. Check the event log for upload results: go to **Statistics > Event Logging**.

## Viewing Access-Log Statistics

Access-log statistics can be viewed from the Management Console or the CLI, although not all statistics you can view in the Management Console are available in the CLI.

You can also view some access log statistics by navigating to **Statistics > Advanced** and clicking **Access Log**. Statistics you can view from **Statistics > Advanced** include:

❐ **Show list of all logs**: The access log manages multiple log objects internally. These are put together as one logical access log file when the file is uploaded.

The show list shows the available internal log objects for easy access. To download part of the access log instead of the whole log file, click on the individual log object shown in the list. The latest log object can be identified by its timestamp.

**Note:** If you have multiple access logs, each access log has its own list of objects.

❐ **Show access log statistics**: The statistics of an individual access log is shown.

❐ **Show statistics of all logs**: The statistics of all the access logs on the system are displayed in a single list.

❐ **Show last N bytes in the log**: The last *N* bytes in the log are shown.

❐ **Show last part of log every time it changes**: A stream of the latest log entries is shown on the page as they are written in the system.

❐ **Show access log tail with optional refresh time**: A refresh from the browser displays the latest log entries.

❐ **Show access log objects**: The statistics of individual access log objects are displayed.

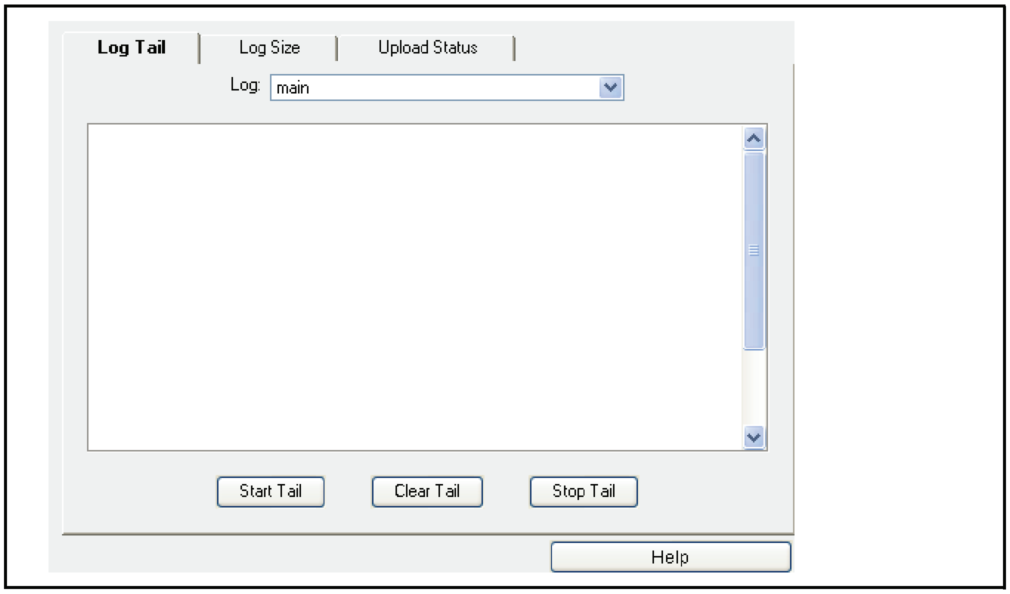
- **Show all access log objects**: The statistics of all access log object are displayed in a single list.

## Viewing the Access Log Tail

This option is not available through the CLI.

**To display the access log tail:**

1. Select **Statistics > Access Logging > Log Tail**.



2. From the **Log** drop-down list, select the log you want to view.
3. Click **Start Tail** to display the access log tail.

   The SG appliance displays a maximum of 500 lines. Entries that pre-date these 500 lines are not displayed.
4. Click **Stop Tail** to stop the display or **Clear Tail** to clear the display.

## Viewing the Log File Size

The Log Size tab displays current log statistics:

- Whether the log is being uploaded (Table 5-1 describes upload statuses)
- The current size of all access log objects
- Disk space usage
- Last modified time
- Estimated size of the access log file, once uploaded

Table 5-1. Log Writing Status Description

| Status | Description |
|---|---|
| `active` | Log writing is active. |
| `active - early upload` | The early upload threshold has been reached. |

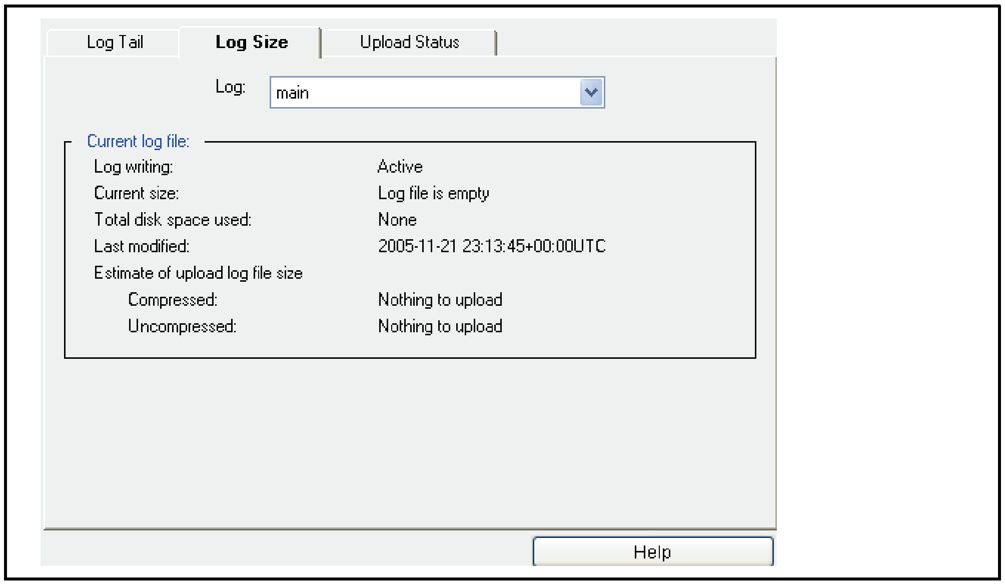
Table 5-1. Log Writing Status Description (Continued)

| | |
|---|---|
| `disabled` | An administrator has disabled logging. |
| `idle` | Log writing is idle. |
| `initializing` | The system is initializing. |
| `shutdown` | The system is shutting down. |
| `stopped` | The access log is full. The maximum log size has been reached. |
| `unknown` | A system error has occurred. |

Estimated compressed size of the uploaded access log and SG appliance access log size might differ during uploading. This occurs because new entries are created during the log upload.

**To view the access log size statistic:**

1. Select **Statistics > Access Logging > Log Size**.



Log: main

Current log file:

| | |
|---|---|
| Log writing: | Active |
| Current size: | Log file is empty |
| Total disk space used: | None |
| Last modified: | 2005-11-21 23:13:45+00:00UTC |
| Estimate of upload log file size | |
| Compressed: | Nothing to upload |
| Uncompressed: | Nothing to upload |

2. From the **Log** drop-down list, select a log to view.

## Viewing Access Logging Status

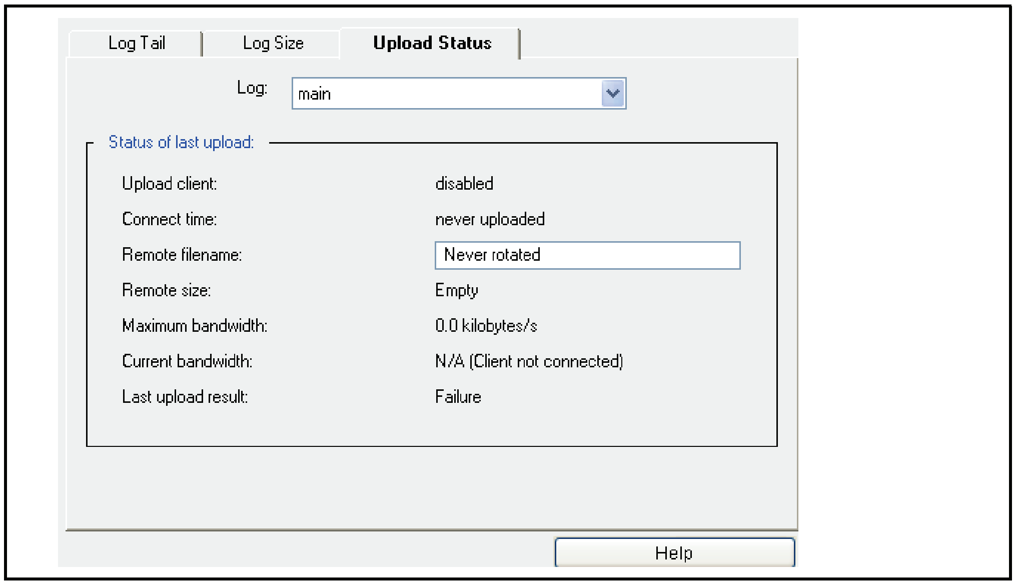
The SGOS software displays the current access logging status on the Management Console. This includes separate status information about:

❐ The writing of access log information to disk

❐ The client the SG appliance uses to upload access log information to the remote server

**To view access logging upload status:**

1. Select **Statistics > Access Logging > Upload Status**.



Log: main

Status of last upload:

| | |
|---|---|
| Upload client: | disabled |
| Connect time: | never uploaded |
| Remote filename: | Never rotated |
| Remote size: | Empty |
| Maximum bandwidth: | 0.0 kilobytes/s |
| Current bandwidth: | N/A (Client not connected) |
| Last upload result: | Failure |

2. Under **Status of Last Upload**, check the appropriate status information displayed in the **Upload client** field.
3. Check the other status information. For information about the status, see the table below.

Table 5-2. Upload Status Information

| Status | Description |
|---|---|
| `Connect time` | The last time a client connection was made or attempted. |
| `Remote filename` | The most recent upload filename. If an access log was encrypted, only the encrypted access log file (the ENC file) displays. |
| `Remote size` | The current size of the upload file. If an access log was encrypted, only the encrypted access log file size (the ENC file) displays. The private key file (the DER file) varies, but is usually about 1 Kb. |
| `Maximum bandwidth` | The maximum bandwidth used in the current or last connection. |
| `Current bandwidth` | The bandwidth used in the last second (available only if currently connected). |
| `Final result` | The result of the last upload attempt (success or failure). This is available only if not connected. |

## Viewing Access-Log Statistics

In the CLI, you can view all access log statistics at once, or you can view the statistics of a specific access log. For details of the meaning of these statistics, see "Viewing the Log File Size" on page 36 and "Viewing Access Logging Status" on page 37.

**To view access logging statistics:**

1. To view the statistics for all access logs at once, enter the following command:
```
SGOS# show access-log statistics
```
2. To view the statistics for a specific access log, enter the following command:
```
SGOS# show access-log statistics log_name
```

The statistics for the access log Main are displayed below as an example:

```
SGOS#(config) show access-log statistics main
Statistics:
Access Log (main) Statistics:
Log Manager Version 3
Log entry lifetime counter:     0
System Status:
        Log manager:                  enabled and running
        Upload client:                disabled
        Log writer:                   idle
        Log reader:                   idle
Log Information:
        Current log size:             0 bytes
        Early upload threshold:       1736 MB
        Maximum log size:             2170 MB
        Max size policy:              stop logging
        Bytes in write buffer :       0
        Tail sockets in use :         0
        Modified time:                2004-08-26 22:10:49+00:00UTC
Next Upload:
        Client type:                  ftp
        Next attempt:                 uploading disabled
        Connect type:                 daily upload
        Connect reason:               regular upload
        Estimated upload size:
          compressed:                  nothing to upload
          uncompressed:                nothing to upload
        Upload format:                gzip
Last Upload Attempt:
        Time:                         never uploaded
        Maximum bandwidth:            0.00 KB/sec
        Result:                       failure
Current/Last Upload File:
        Remote filename:              Never rotated
        Remote size:                  0 bytes
Using Access Logging with Policy Rules
```

After configuration is complete, you must create rules to manage the access logs you set up. You can create rules through the Visual Policy Manager module of the Management Console, or you can use Content Policy Language (CPL) directly (refer to *Volume 11: Blue Coat SG Appliance Content Policy Language Guide*).

Actions you can do to manage access logging:

- Reset logging to its default
- Disable all logging
- Add logging to a log file
- Disable logging to a log file
- Override specific access-log fields

You can also set the list of logs to be used, but you must use CPL to create this action. It is not available through VPM.

The first two actions—reset logging to its default and disable all logging—are referred to as constant actions, just like the allow/deny actions. Select only one per rule.

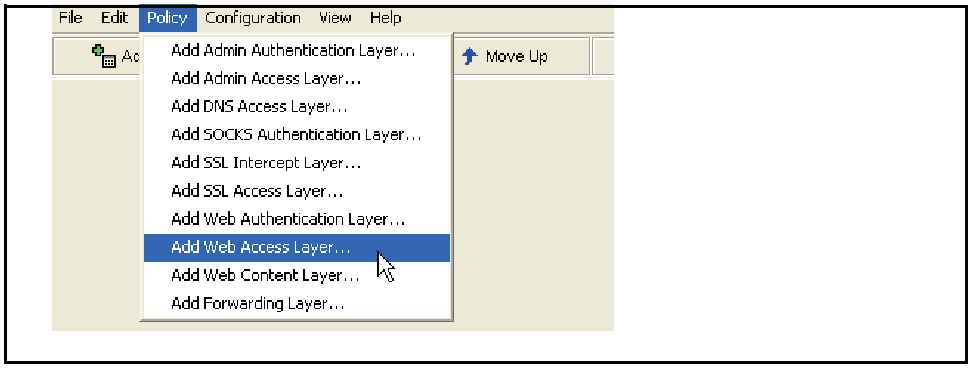
All of the actions are allowed in all layers. If you use VPM, the access-logging actions display in the VPM policy; if you use CPL, you can put the actions into any file, but Blue Coat recommends you use the Local file.

## Example: Using VPM to Prevent Logging of Entries Matching a Source IP

Complete the following steps to prevent a source IP address from being logged.

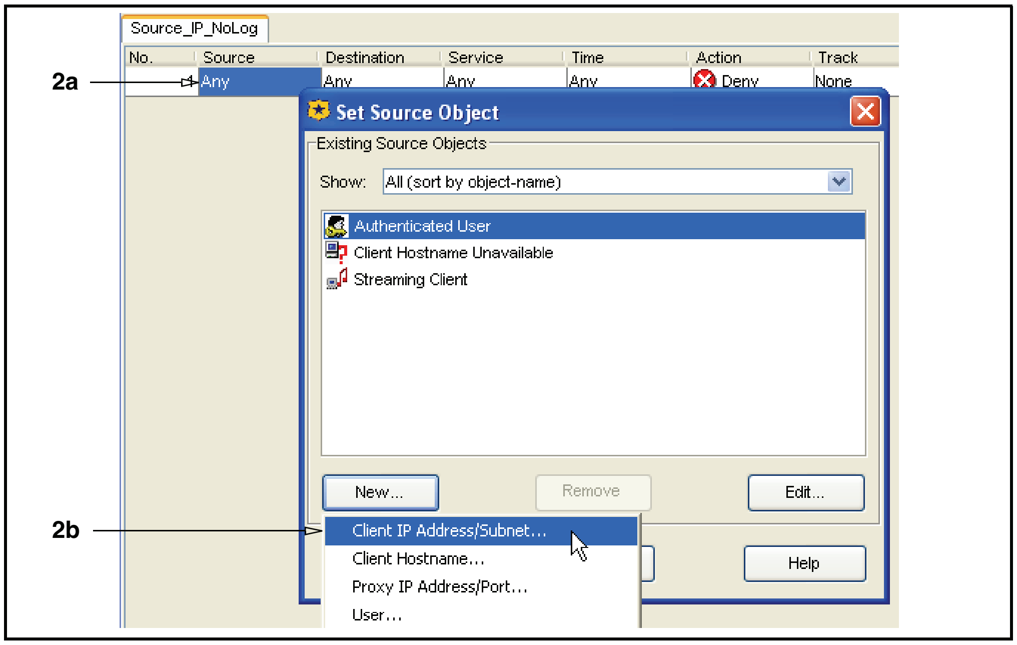
**To prevent a source IP address from being logged:**

1. Create a Web Access Layer:
   a. Select **Configuration > Policy > Visual Policy Manager**; click **Launch**.



   b. In the VPM, select **Policy > Add Web Access Layer**.
   c. Enter a layer name into the dialog that appears and click **OK**.

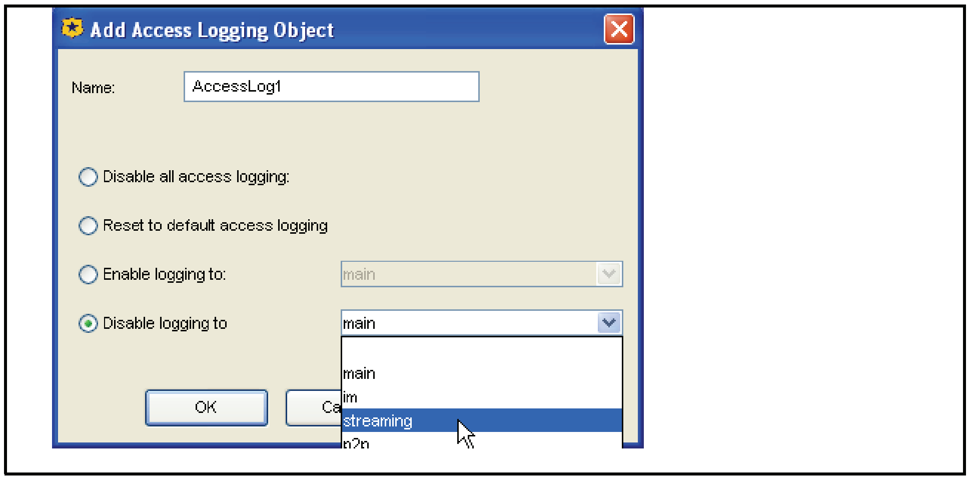
2. Add a Source object:



   a. Right click on the item in the **Source** column; select **Set**.
   b. Click **New**; select **Client IP Address/Subnet**.

3. Enter an IP address or Subnet Mask in the dialog that appears and click **Add**; click **Close** (or add additional addresses and then click **Close**); click **OK**.

4. Add an Action object to this rule:
   a. Right-click on the item in the **Action** column; select **Set**.
   b. Click **New** in the Set Action Object dialog that appears; select **Modify Access Logging**.



Add Access Logging Object

Name: AccessLog1

( ) Disable all access logging:

( ) Reset to default access logging

( ) Enable logging to: main

(•) Disable logging to main

   c. To disable a particular log, click **Disable logging to** and select that log from the drop-down list; to disable all access logging, click **Disable all access logging**.

5. Click **OK**; click **OK** again; close the VPM window and click **Yes** in the dialog to save your changes.

# Appendix A:  Glossary

| Term | Description |
|---|---|
| ADN Optimize Attribute | Controls whether to optimize bandwidth usage when connecting upstream using an ADN tunnel. |
| Asynchronous Adaptive Refresh (AAR) | This allows the Proxy*SG* to keep cached objects as fresh as possible, thus reducing response times. The AAR algorithm allows HTTP proxy to manage cached objects based on their rate of change and popularity: an object that changes frequently and/or is requested frequently is more eligible for asynchronous refresh compared to an object with a lower rate of change and/or popularity. |
| Asynchronous Refresh Activity | Refresh activity that does not wait for a request to occur, but that occurs *asynchronously* from the request. |
| Attributes (Service) | The service attributes define the parameters, such as explicit or transparent, cipher suite, and certificate verification, that the SG appliance uses for a particular service. . |
| Authenticate-401 Attribute | All transparent and explicit requests received on the port always use transparent authentication (cookie or IP, depending on the configuration). This is especially useful to force transparent proxy authentication in some proxy-chaining scenarios |
| authentication | The process of identifying a specific user. |
| authorization | The permissions given to a specific user. |
| Bandwidth Gain | A measure of the difference in client-side and server-side Internet traffic expressed in relation to server-side Internet traffic. It is managed in two ways: you can enable or disable bandwidth gain mode or you can select the Bandwidth Gain profile (this also enables bandwidth gain mode).. |
| Bandwidth Class | A defined unit of bandwidth allocation. An administrator uses bandwidth classes to allocate bandwidth to a particular type of traffic flowing through the SG appliance. |
| Bandwidth Class Hierarchy | Bandwidth classes can be grouped together in a class hierarchy, which is a tree structure that specifies the relationship among different classes. You create a hierarchy by creating at least one parent class and assigning other classes to be its children. |
| Bandwidth Policy | The set of rules that you define in the policy layer to identify and classify the traffic in the SG appliance, using the bandwidth classes that you create. You must use policy (through either VPM or CPL) in order to manage bandwidth. |
| Bypass Lists | The bypass list allows you to exempt IP addresses from being proxied by the SG appliance. The bypass list allows either <All> or a specific IP prefix entry for both the client and server columns. Both UDP and TCP traffic is automatically exempted. |

| Term | Description |
|---|---|
| Byte-Range Support | The ability of the Proxy*SG* to respond to byte-range requests (requests with a `Range:` HTTP header). |
| Cache-hit | An object that is in the Proxy*SG* and can be retrieved when an end user requests the information. |
| Cache-miss | An object that can be stored but has never been requested before; it was not in the Proxy*SG* to start, so it must be brought in and stored there as a side effect of processing the end-user's request. If the object is cacheable, it is stored and served the next time it is requested. |
| Child Class (Bandwidth Gain) | The child of a parent class is dependent upon that parent class for available bandwidth (they share the bandwidth in proportion to their minimum/maximum bandwidth values and priority levels). A child class with siblings (classes with the same parent class) shares bandwidth with those siblings in the same manner. |
| Client consent certificates | A certificate that indicates acceptance or denial of consent to decrypt an end user's HTTPS request. |
| Compression | An algorithm that reduces a file's size but does not lose any data. The ability to compress or decompress objects in the cache is based on policies you create. Compression can have a huge performance benefit, and it can be customized based on the needs of your environment: Whether CPU is more expensive (the default assumption), server-side bandwidth is more expensive, or whether client-side bandwidth is more expensive. |
| Default Proxy Listener | See " Proxy Service (Default)" . |
| Detect Protocol Attribute | Detects the protocol being used. Protocols that can be detected include:<br>HTTP, P2P (eDonkey, BitTorrent, FastTrack, Gnutella), SSL, and Endpoint Mapper. |
| Directives | Directives are commands that can be used in installable lists to configure forwarding. See also *forwarding Configuration*. |
| Display Filter | The display filter is a drop-down list at the top of the Proxy Services pane that allows you to view the created proxy services by service name or action. |
| Early Intercept Attribute | Controls whether the proxy responds to client TCP connection requests before connecting to the upstream server. When early intercept is disabled, the proxy delays responding to the client until after it has attempted to contact the server. |
| Emulated Certificates | Certificates that are presented to the user by ProxySG when intercepting HTTPS requests. Blue Coat emulates the certificate from the server and signs it, copying the subjectName and expiration. The original certificate is used between the Proxy*SG* and the server. |
| ELFF-compatible format | A log type defined by the W3C that is general enough to be used with any protocol. |
| Encrypted Log | A log is encrypted using an external certificate associated with a private key. Encrypted logs can only be decrypted by someone with access to the private key. The private key is not accessible to the SG appliance. |

| Term | Description |
|---|---|
| explicit proxy | A configuration in which the browser is explicitly configured to communicate with the proxy server for access to content.<br>This is the default for the SG appliance, and requires configuration for both browser and the interface card. |
| Fail Open/Closed | Failing open or closed applies to forwarding hosts and groups and SOCKS gateways. Fail Open/Closed applies when the health checks are showing sick for each forwarding or SOCKS gateway target in the applicable fail-over sequence. If no systems are healthy, the SG appliance fails open or closed, depending on the configuration. If closed, the connection attempt simply fails.<br>If open, an attempt is made to connect without using any forwarding target (or SOCKS gateway). Fail open is usually a security risk; fail closed is the default if no setting is specified. |
| Forwarding Configuration | Forwarding can be configured through the CLI or through adding directives to a text file and installing it as an installable list. Each of these methods (the CLI or using directives) is equal. You cannot use the Management Console to configure forwarding. |
| Forwarding Host | Upstream Web servers or proxies. |
| forward proxy | A proxy server deployed close to the clients and used to access many servers. A forward proxy can be explicit or transparent. |
| Freshness | A percentage that reflects the objects in the Proxy*SG* cache that are expected to be fresh; that is, the content of those objects is expected to be identical to that on the OCS (origin content server). |
| Gateway | A device that serves as entrance and exit into a communications network. |
| Global Default Settings | You can configure settings for all forwarding hosts and groups. These are called the global defaults. You can also configure private settings for each individual forwarding host or group. Individual settings override the global defaults. |
| FTP | See Native FTP; Web FTP. |
| Host Affinity | Host affinity is the attempt to direct multiple connections by a single user to the same group member. Host affinity is closely tied to load balancing behavior; both should configured if load balancing is important. |
| Host Affinity Timeout | The host affinity timeout determines how long a user remains idle before the connection is closed. The timeout value checks the user's IP address, SSL ID, or cookie in the host affinity table. |
| Inbound Traffic (Bandwidth Gain) | Network packets flowing into the SG appliance. Inbound traffic mainly consists of the following:<br>• Server inbound: Packets originating at the origin content server (OCS) and sent to the SG appliance to load a Web object.<br>• Client inbound: Packets originating at the client and sent to the SG appliance for Web requests. |

| Term | Description |
|---|---|
| Installable Lists | Installable lists, comprised of directives, can be placed onto the SG appliance in one of several methods: through creating the list through the SG text editor, by placing the list at an accessible URL, or by downloading the directives file from the local system. |
| Integrated Host Timeout | An integrated host is an Origin Content Server (OCS) that has been added to the health check list. The host, added through the `integrate_new_hosts` property, ages out of the integrated host table after being idle for the specified time. The default is 60 minutes. |
| IP Reflection | Determines how the client IP address is presented to the origin server for explicitly proxied requests. All proxy services contain a reflect-ip attribute, which enables or disables sending of client's IP address instead of the SG's IP address. |
| Issuer keyring | The keyring that is used by the SG appliance to sign emulated certificates. The keyring is configured on the appliance and managed through policy. |
| Listener | The service that is listening on a specific port. A listener can be identified by any destination IP/subnet and port range. Multiple listeners can be added to each service. |
| Load Balancing | The ability to share traffic requests among multiple upstream targets. Two methods can be used to balance the load among systems: `least-connections` or `round-robin.` |
| Log Facility | A separate log that contains a single logical file and supports a single log format. It also contains the file's configuration and upload schedule information as well as other configurable information such as how often to rotate (switch to a new log) the logs at the destination, any passwords needed, and the point at which the facility can be uploaded. |
| Log Format | The type of log that is used: NCSA/Common, SQUID, ELFF, SurfControl, or Websense.<br>The proprietary log types each have a corresponding pre-defined log format that has been set up to produce exactly that type of log (these logs cannot be edited). In addition, a number of other ELFF type log formats are also pre-defined (im, main, p2p, ssl, streaming). These can be edited, but they start out with a useful set of log fields for logging particular protocols understood by the SG appliance. It is also possible to create new log formats of type ELFF or Custom which can contain any desired combination of log fields. |
| Log Tail: | The access log tail shows the log entries as they get logged. With high traffic on the SG appliance, not all access log entries are necessarily displayed. However, you can view all access log information after uploading the log. |
| Maximum Object Size | The maximum object size stored in the Proxy*SG*. All objects retrieved that are greater than the maximum size are delivered to the client but are not stored in the Proxy*SG*. |
| NCSA common log format | A log type that contains only basic HTTP access information. |

| Term | Description |
|---|---|
| Negative Responses | An error response received from the OCS when a page or image is requested. If the Proxy*SG* is configured to cache such negative responses, it returns that response in subsequent requests for that page or image for the specified number of minutes. If it is not configured, which is the default, the Proxy*SG* attempts to retrieve the page or image every time it is requested. |
| Native FTP | Native FTP involves the client connecting (either explicitly or transparently) using the FTP protocol; the SG appliance then connects upstream through FTP (if necessary). |
| Outbound Traffic (Bandwidth Gain) | Network packets flowing out of the SG appliance. Outbound traffic mainly consists of the following:<br>• Client outbound: Packets sent to the client in response to a Web request.<br>• Server outbound: Packets sent to an OCS or upstream proxy to request a service. |
| Origin Content Server (OCS) | |
| Parent Class (Bandwidth Gain) | A class with at least one child. The parent class must share its bandwidth with its child classes in proportion to the minimum/maximum bandwidth values or priority levels. |
| PASV | Passive Mode Data Connections. Data connections initiated by an FTP client to an FTP server. |
| proxy | Caches content, filters traffic, monitors Internet and intranet resource usage, blocks specific Internet and intranet resources for individuals or groups, and enhances the quality of Internet or intranet user experiences.<br>A proxy can also serve as an intermediary between a Web client and a Web server and can require authentication to allow identity based policy and logging for the client.<br>The rules used to authenticate a client are based on the policies you create on the SG appliance, which can reference an existing security infrastructure—LDAP, RADIUS, IWA, and the like. |
| Proxy Service | The proxy service defines the ports, as well as other attributes. that are used by the proxies associated with the service. |
| Proxy Service (Default) | The default proxy service is a service that intercepts all traffic not otherwise intercepted by other listeners. It only has one listener whose action can be set to bypass or intercept. No new listeners can be added to the default proxy service, and the default listener and service cannot be deleted. Service attributes can be changed. |
| realms | A realm is a named collection of information about users and groups. The name is referenced in policy to control authentication and authorization of users for access to Blue Coat Systems SG services. Multiple authentication realms can be used on a single SG appliance. Realm services include IWA, LDAP, Local, and RADIUS. |
| Reflect Client IP Attribute | Enables the sending of the client's IP address instead of the SG's IP address to the upstream server. If you are using an Application Delivery Network (ADN), this setting is enforced on the concentrator proxy through the **Configuration>App. Delivery Network>Tunneling** tab. |

| Term | Description |
|---|---|
| Refresh Bandwidth | The amount of bandwidth used to keep stored objects fresh. By default, the Proxy*SG* is set to manage refresh bandwidth automatically. You can configure refresh bandwidth yourself, although Blue Coat does not recommend this. |
| reverse proxy | A proxy that acts as a front-end to a small number of pre-defined servers, typically to improve performance. Many clients can use it to access the small number of predefined servers. |
| rotate logs | When you rotate a log, the old log is no longer appended to the existing log, and a new log is created. All the facility information (headers for passwords, access log type, and so forth), is re-sent at the beginning of the new upload.<br>If you're using Reporter (or anything that doesn't understand the concept of "file," such as streaming) the upload connection is broken and then re-started, and, again, the headers are re-sent. |
| serial console | A device that allows you to connect to the SG appliance when it is otherwise unreachable, without using the network. It can be used to administer the SG appliance through the CLI. You must use the CLI to use a serial console.<br>Anyone with access to the serial console can change the administrative access controls, so physical security of the serial console is critical. |
| Server Certificate Categories | The hostname in a server certificate can be categorized by BCWF or another content filtering vendor to fit into categories such as banking, finance, sports. |
| Sibling Class (Bandwidth Gain) | A bandwidth class with the same parent class as another class. |
| SOCKS Proxy | A generic way to proxy TCP and UDP protocols. The SG appliance supports both SOCKSv4/4a and SOCKSv5; however, because of increased username and password authentication capabilities and compression support, Blue Coat recommends that you use SOCKS v5.. |
| SmartReporter log type | A proprietary ELFF log type that is compatible with the SmartFilter SmartReporter tool. |
| Split proxy | Employs co-operative processing at the branch and the core to implement functionality that is not possible in a standalone proxy. Examples of split proxies include :<br>Mapi Proxy<br>SSL Proxy |
| SQUID-compatible format | A log type that was designed for cache statistics. |
| SSL | A standard protocol for secure communication over the network. Blue Coat recommends using this protocol to protect sensitive information. |
| SSL Interception | Decrypting SSL connections. |
| SSL Proxy | A proxy that can be used for any SSL traffic (HTTPS or not), in either forward or reverse proxy mode. |

| Term | Description |
|---|---|
| static routes | A manually-configured route that specifies the transmission path a packet must follow, based on the packet's destination address. A static route specifies a transmission path to another network. |
| SurfControl log type | A proprietary log type that is compatible with the SurfControl reporter tool. The SurfControl log format includes fully-qualified usernames when an NTLM realm provides authentication. The simple name is used for all other realm types. |
| Traffic Flow (Bandwidth Gain) | Also referred to as *flow.* A set of packets belonging to the same TCP/UDP connection that terminate at, originate at, or flow through the SG appliance. A single request from a client involves two separate connections. One of them is from the client to the SG appliance, and the other is from the SG appliance to the OCS. Within each of these connections, traffic flows in two directions—in one direction, packets flow out of the SG appliance (outbound traffic), and in the other direction, packets flow into the SG (inbound traffic). Connections can come from the client or the server. Thus, traffic can be classified into one of four types:<br>• Server inbound<br>• Server outbound<br>• Client inbound<br>• Client outbound<br>These four traffic flows represent each of the four combinations described above. Each flow represents a single direction from a single connection. |
| transparent proxy | A configuration in which traffic is redirected to the SG appliance without the knowledge of the client browser. No configuration is required on the browser, but network configuration, such as an L4 switch or a WCCP-compliant router, is required. |
| Variants | Objects that are stored in the cache in various forms: the original form, fetched from the OCS; the transformed (compressed or uncompressed) form (if compression is used). If a required compression variant is not available, then one might be created upon a cache-hit. (Note: policy-based content transformations are not stored in the Proxy*SG*.) |
| Web FTP | Web FTP is used when a client connects in explicit mode using HTTP and accesses an ftp:// URL. The SG appliance translates the HTTP request into an FTP request for the OCS (if the content is not already cached), and then translates the FTP response with the file contents into an HTTP response for the client. |
| *Websense* log type | A proprietary log type that is compatible with the Websense reporter tool. |

| Term | Description |
|---|---|
| Wildcard Services | When multiple non-wildcard services are created on a port, all of them must be of the same service type (a wildcard service is one that is listening for that port on all IP addresses). If you have multiple IP addresses and you specify IP addresses for a port service, you cannot specify a different protocol if you define the same port on another IP address. For example, if you define HTTP port 80 on one IP address, you can only use the HTTP protocol on port 80 for other IP addresses.<br>Also note that wildcard services and non-wildcard services cannot both exist at the same time on a given port.<br>For all service types except HTTPS, a specific listener cannot be posted on a port if the same port has a wildcard listener of any service type already present. |

# Appendix B: Access Log Formats

The SG appliance can create access logs in one of the following formats:

- "Custom or W3C ELFF Format"
- "SQUID-Compatible Format" on page 53
- "NCSA Common Access Log Format" on page 55

ELFF is a log format defined by the W3C that contains information about Windows Media and RealProxy logs.

The SG appliance can create access logs with any one of six formats. Four of the six are reserved formats and cannot be configured. However, you can create additional logs using custom or ELFF format strings.

When using an ELFF or custom format, a blank field is represented by a dash character. When using the SQUID or NCSA log format, a blank field is represented according to the standard of the format.

## Custom or W3C ELFF Format

The W3C Extended Log File Format (ELFF) is a subset of the Blue Coat Systems format. The ELFF format is specified as a series of space delimited fields. Each field is described using a text string. The types of fields are described in Table 7-1.

Table 7-1. Field Types

| Field Type | Description |
|---|---|
| `Identifier` | A type unrelated to a specific party, such as date and time. |
| `prefix-identifier` | Describes information related to a party or a transfer, such as `c-ip` (client's IP) or `sc-bytes` (how many bytes were sent from the server to the client) |
| `prefix (header)` | Describes a header data field. The valid prefixes are:<br><br>`c` = Client     `cs` = Client to Server<br>`s` = Server     `sc` = Server to Client<br>`r` = Remote     `rs` = Remote to Server<br>`sr` = Server to Remote |

ELFF formats are created by selecting a corresponding custom log format using the table below. Unlike the Blue Coat custom format, ELFF does not support character strings and require a space between fields.

Selecting the ELFF format does the following:

- Puts one or more W3C headers into the log file. Each header contains the following lines:

```
#Software: SGOS x.x.x
#Version: 1.0
#Date: 2002-06-06 12:12:34
#Fields: date time cs-ip…
```

- Changes all spaces within fields to `+` or `%20`. The ELFF standard requires that spaces only be present between fields.

ELFF formats are described in Table 7-2.

Table 7-2. Blue Coat Custom Format and Extended Log File Format

| Blue Coat Custom Format | Extended Log File Format | Description |
|---|---|---|
| space character | N/A | Multiple consecutive spaces are compressed to a single space. |
| % | - | Denotes an expansion field. |
| %% | - | Denotes '%' character. |
| %a | c-ip | IP address of the client |
| %b | sc-bytes | Number of bytes sent from appliance to client |
| %c | rs(Content-Type) | Response header: Content-Type |
| %d | s-supplier-name | Hostname of the upstream host (not available for a cache hit) |
| %e | time-taken | Time taken (in milliseconds) to process the request |
| %f | sc-filter-category | Content filtering category of the request URL |
| %g | timestamp | Unix type timestamp |
| %h | c-dns | Hostname of the client (uses the client's IP address to avoid reverse DNS) |
| %i | cs-uri | The 'log' URL. |
| %j | - | [Not used.] |
| %k | - | [Not used.] |
| %l | x-bluecoat-special-empty | Resolves to an empty string |
| %m | cs-method | Request method used from client to appliance |
| %n | - | [Not used.] |
| %o | - | [Not used.] |
| %p | r-port | Port from the outbound server URL |
| %q | - | [Not used.] |
| %r | cs-request-line | First line of the client's request |
| %s | sc-status | Protocol status code from appliance to client |
| %t | gmttime | GMT date and time of the user request in format: [DD/MM/YYYY:hh:mm:ss GMT] |
| %u | cs-user | Qualified username for NTLM. Relative username for other protocols |
| %v | cs-host | Hostname from the client's request URL. If URL rewrite policies are used, this field's value is derived from the 'log' URL |
| %w | s-action | What type of action did the Appliance take to process this request. |
| %x | date | GMT Date in YYYY-MM-DD format |
| %y | time | GMT time in HH:MM:SS format |
| %z | s-icap-status | ICAP response status |
| %A | cs(User-Agent) | Request header: User-Agent |
| %B | cs-bytes | Number of bytes sent from client to appliance |
| %C | cs(Cookie) | Request header: Cookie |

Table 7-2. Blue Coat Custom Format and Extended Log File Format (Continued)

| Blue Coat Custom Format | Extended Log File Format | Description |
|---|---|---|
| %D | s-supplier-ip | IP address used to contact the upstream host (not available for a cache hit) |
| %E | - | [Not used.] |
| %F | - | [Not used.] |
| %G | - | [Not used.] |
| %H | s-hierarchy | How and where the object was retrieved in the cache hierarchy. |
| %I | s-ip | IP address of the appliance on which the client established its connection |
| %J | - | [Not used.] |
| %K | - | [Not used.] |
| %L | localtime | Local date and time of the user request in format: [DD/MMM/YYYY:hh:mm:ss +nnnn] |
| %M | - | [Not used.] |
| %N | s-computername | Configured name of the appliance |
| %O | - | [Not used.] |
| %P | s-port | Port of the appliance on which the client established its connection |
| %Q | cs-uri-query | Query from the 'log' URL. |
| %R | cs(Referer) | Request header: Referer |
| %S | s-sitename | The service type used to process the transaction |
| %T | duration | Time taken (in seconds) to process the request |
| %U | cs-uri-path | Path from the 'log' URL. Does not include query. |
| %V | cs-version | Protocol and version from the client's request, e.g. HTTP/1.1 |
| %W | sc-filter-result | Content filtering result: Denied, Proxied or Observed |
| %X | cs(X-Forwarded-For) | Request header: X-Forwarded-For |
| %Y | - | [Not used.] |
| %Z | s-icap-info | ICAP response information |

### Example Access Log Formats

```
Squid log format: %g %e %a %w/%s %b %m %i %u %H/%d %c
NCSA common log format: %h %l %u %t “%r” %s %b
NCSA extended log format: %h %l %u %L "%r" %s %b "%R" "%A"
Microsoft IIS format: %a, -, %x, %y, %S, %N, %I, %e, %b, %B, %s, 0, %m,
%U, -
```

The Blue Coat custom format allows any combination of characters and format fields. Multiple spaces are compressed to a single space in the actual access log. You can also enter a string, such as `My default is %d`. The SG appliance goes through such strings and finds the relevant information. In this case, that information is `%d`.

## SQUID-Compatible Format

The SQUID-compatible format contains one line for each request. For SQUID-1.1, the format is:

```
time elapsed remotehost code/status bytes method URL rfc931
peerstatus/peerhost type
```

For SQUID-2, the columns stay the same, though the content within might change a little.

## Action Field Values

Table 7-3 describes the possible values for the action field.

Table 7-3. Action Field Values

| Value | Description |
|---|---|
| `ACCELERATED` | (SOCKS only) The request was handed to the appropriate protocol agent for handling. |
| `ALLOWED` | An FTP method (other than the data transfer method) is successful. |
| `DENIED` | Policy denies a method. |
| `FAILED` | An error or failure occurred. |
| `LICENSE_EXPIRED` | (SOCKS only) The request could not be handled because the associated license has expired. |
| `TUNNELED` | Successful data transfer operation. |
| `TCP_` | Refers to requests on the HTTP port. |
| `TCP_AUTH_HIT` | The requested object requires upstream authentication, and was served from the cache. |
| `TCP_AUTH_MISS` | The requested object requires upstream authentication, and was not served from the cache. This is part of CAD (Cached Authenticated Data). |
| `TCP_AUTH_REDIRECT` | The client was redirected to another URL for authentication. |
| `TCP_CLIENT_REFRESH` | The client forces a revalidation with the origin server with a `Pragma: no-cache`. If the server returns `304 Not Modified`, this appears in the `Statistics:Efficiency` file as `In Cache, verified Fresh`. |
| `TCP_DENIED` | Access to the requested object was denied by a filter. |
| `TCP_ERR_MISS` | An error occurred while retrieving the object from the origin server. |
| `TCP_HIT` | A valid copy of the requested object was in the cache. |
| `TCP_LOOP` | The current connection is dropped because the upstream connection would result in a looped connection. |
| `TCP_MEM_HIT` | The requested object was, in its entirety, in RAM. |
| `TCP_MISS` | The requested object was not in the cache. |
| `TCP_NC_MISS` | The object returned from the origin server was non-cacheable. |
| `TCP_PARTIAL_MISS` | The object is in the cache, but retrieval from the origin server is in progress. |
| `TCP_POLICY_REDIRECT` | The client was redirected to another URL due to policy. |

Table 7-3. Action Field Values (Continued)

| Value | Description |
|---|---|
| `TCP_REFRESH_HIT` | A GIMS request to the server was forced and the response was `304 Not Modified`, this appears in the `Statistics:Efficiency` file as `In Cache, verified Fresh`. |
| `TCP_REFRESH_MISS` | A GIMS request to the server was forced and new content was returned. |
| `TCP_RESCAN_HIT` | The requested object was found in the cache but was rescanned because the virus-scanner-tag-id in the object was different from the current scanner tag. |
| `TCP_SPLASHED` | The user was redirected to a splash page. |
| `TCP_SWAPFAIL` | The object was believed to be in the cache, but could not be accessed. |
| `TCP_TUNNELED` | The CONNECT method was used to tunnel this request (generally proxied HTTPS). |
| `UDP_` | Refers to requests on the ICP port (3130). |
| `UDP_DENIED` | Access was denied for this request. |
| `UDP_HIT` | A valid copy of the requested object was in the cache. This value is also used with ICP queries. |
| `UDP_INVALID` | The ICP request was corrupt, short, or otherwise unintelligible. |
| `UDP_MISS` | The requested object was not in the cache. This value is also used with ICP queries. |
| `UDP_MISS_NOFETCH` | An ICP request was made to this cache for an object not in the cache. The requestor was informed that it could not use this cache as a parent to retrieve the object. (This is not supported at this time.) |
| `UDP_OBJ` | An ICP request was made to this cache for an object that was in cache, and the object was returned through UDP. (This is not supported at this time. This functionality is deprecated in the current ICP specification.) |

## NCSA Common Access Log Format

The common log format contains one line for each request. The format of each log entry is shown below:

```
remotehost rfc931 authuser [date] “request” status bytes
```

Each field is described in Table 7-4.

Table 7-4. Log Entry Fields

| Field Name | Description |
|---|---|
| `remotehost` | DNS hostname or IP address of remote server. |
| `rfc931` | The remote log name of the user. This field is always —. |
| `authuser` | The username as which the user has authenticated himself. |
| `[date]` | Date and time of the request. |
| `“request”` | The request line exactly as it came from the client. |

Table 7-4. Log Entry Fields (Continued)

| | |
|---|---|
| `status` | The HTTP status code returned to the client. |
| `bytes` | The content length of the document transferred. |

## Access Log Filename Formats

Table 7-5 details the specifiers for the access log upload filenames.

Table 7-5. Specifies for Access Log Upload Filenames

| Specifier | Description |
|---|---|
| `%%` | Percent sign. |
| `%a` | Abbreviated weekday name. |
| `%A` | Full weekday name. |
| `%b` | Abbreviated month name. |
| `%B` | Full month name. |
| `%c` | The certificate name used for encrypting the log file (expands to nothing in non-encrypted case). |
| `%C` | The SG appliance name. |
| `%d` | Day of month as decimal number (`01 - 31`). |
| `%f` | The log name. |
| `%H` | Hour in 24-hour format (`00 - 23`). |
| `%i` | First IP address of the SG appliance, displayed in `x_x_x_x` format, with leading zeros removed. |
| `%I` | Hour in 12-hour format (`01 - 12`). |
| `%j` | Day of year as decimal number (`001 - 366`). |
| `%l` | The fourth part of the SG appliance's IP address, using three digits (`001.002.003.`**`004`**) |
| `%m` | Month as decimal number (`01 - 12`). |
| `%M` | Minute as decimal number (`00 - 59`). |
| `%p` | Current locale's A.M./P.M. indicator for 12-hour clock. |
| `%S` | Second as decimal number (`00 - 59`). |
| `%U` | Week of year as decimal number, with Sunday as first day of week (`00 - 53`). |
| `%w` | Weekday as decimal number (`0 - 6`; Sunday is `0`). |
| `%W` | Week of year as decimal number, with Monday as first day of week (`00 - 53`). |
| `%y` | Year without century, as decimal number (`00 - 99`). |
| `%Y` | Year with century, as decimal number. |
| `%z, %Z` | Time-zone name or abbreviation; no characters if time zone is unknown. |

## Fields Available for Creating Access Log Formats

The following table lists all fields available for creating access log formats. When creating an ELFF format, you must use the values from the ELFF column. When creating a custom format, you can use values from the ELFF, CPL, or custom column.

Table 7-6. Access Log Formats

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| **Category: bytes** | | | |
| cs-bodylength | | | Number of bytes in the body (excludes header) sent from client to appliance |
| cs-bytes | | %B | Number of bytes sent from client to appliance |
| cs-headerlength | | | Number of bytes in the header sent from client to appliance |
| rs-bodylength | | | Number of bytes in the body (excludes header) sent from upstream host to appliance |
| rs-bytes | | | Number of bytes sent from upstream host to appliance |
| rs-headerlength | | | Number of bytes in the header sent from upstream host to appliance |
| sc-bodylength | | | Number of bytes in the body (excludes header) sent from appliance to client |
| sc-bytes | | %b | Number of bytes sent from appliance to client |
| sc-headerlength | | | Number of bytes in the header sent from appliance to client |
| sr-bodylength | | | Number of bytes in the body (excludes header) sent from appliance to upstream host |
| sr-bytes | | | Number of bytes sent from appliance to upstream host |
| sr-headerlength | | | Number of bytes in the header sent from appliance to upstream host |
| | | | |
| **Category: cifs** | | | |
| x-cifs-bytes-written | | | Total number of bytes written to the associated resource |
| x-cifs-client-bytes-read | | | Total number of bytes read by CIFS client from the associated resource |
| x-cifs-client-read-operations | | | Total number of read operations issued by the CIFS client for the associated resource |
| x-cifs-client-other-operations | | | Total number of non read/write operations issued by the CIFS client for the associated resource |
| x-cifs-client-write-operations | | | Total number of write operations issued by the CIFS client for the associated resource |
| x-cifs-dos-error-class | | | DOS error class generated by server, in hexadecimal |
| x-cifs-dos-error-code | | | DOS error code generated by server, in hexadecimal |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-cifs-error-code | | | Error code generated by server |
| x-cifs-fid | | | ID representing a CIFS resource |
| x-cifs-file-size | | | Size in bytes of CIFS resource |
| x-cifs-file-type | | | Type of CIFS resource |
| x-cifs-method | | | The method associated with the CIFS request |
| x-cifs-nt-error-code | | | NT error code generated by server, in hexadecimal |
| x-cifs-orig-path | | | Original path name of resource to be renamed |
| x-cifs-orig-unc-path | | | UNC path of original path name of resource to be renamed |
| x-cifs-path | | | CIFS resource name as specified in the UNC path |
| x-cifs-server | | | CIFS server as specified in the UNC path |
| x-cifs-server-bytes-read | | | Total number of bytes read by CIFS server from the associated resource |
| x-cifs-server-operations | | | Total number of operations issued to the CIFS server for the associated resource |
| x-cifs-share | | | CIFS share name as specified in the UNC path |
| x-cifs-tid | | | ID representing instance of an authenticated connection to server resource |
| x-cifs-uid | | | ID representing an authenticated user instance |
| x-cifs-unc-path | | | CIFS path of form \\\\server\\share\\path where path may be empty |
| | | | |
| **Category: connection** | | | |
| cs-ip | proxy.address | | IP address of the destination of the client's connection |
| c-connect-type | | | The type of connection made by the client to the appliance -- 'Transparent' or 'Explicit' |
| c-dns | | %h | Hostname of the client (uses the client's IP address to avoid reverse DNS) |
| x-cs-dns | client.host | | The hostname of the client obtained through reverse DNS. |
| c-ip | client.address | %a | IP address of the client |
| c-port | | | Source port used by the client |
| x-cs-netbios-computer-name | netbios.computer-name | | The NetBIOS name of the computer. This is an empty string if the query fails or the name is not reported. When using the $(netbios.*) substitutions to generate the username, the client machines must react to a NetBIOS over TCP/IP node status query. |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-cs-netbios-computer-domain | netbios.computer-domain | | The name of the domain to which the computer belongs. This is an empty string if the query fails or the name is not reported. When using the $(netbios.*) substitutions to generate the username, the client machines must react to a NetBIOS over TCP/IP node status query. |
| x-cs-netbios-messenger-username | netbios.messenger-username | | The name of the logged-in user. This is an empty string if the query fails or the name is not reported. It is also empty there is more than one logged-in user. When using the $(netbios.*) substitutions to generate the username, the client machines must react to a NetBIOS over TCP/IP node status query. |
| x-cs-netbios-messenger-usernames | netbios.messenger-usernames | | A comma-separated list of the all the messenger usernames reported by the target computer. This is an empty string if the query fails, or no names are reported. When using the $(netbios.*) substitutions to generate the username, the client machines must react to a NetBIOS over TCP/IP node status query. |
| x-cs-session-username | session.username | | The username associated with this session as reported by RADIUS accounting. This is an empty string if no session is known. |
| x-cs-connection-negotiated-cipher | client.connection.negotiated_cipher | | OpenSSL cipher suite negotiated for the client connection |
| x-cs-connection-negotiated-cipher-strength | client.connection.negotiated_cipher.strength | | Strength of the OpenSSL cipher suite negotiated for the client connection |
| x-cs-connection-negotiated-cipher-size | | | Ciphersize of the OpenSSL cipher suite negotiated for the client connection |
| x-cs-connection-negotiated-ssl-version | client.connection.negotiated_ssl_version | | Version of the SSL protocol negotiated for the client connection |
| r-dns | | | Hostname from the outbound server URL |
| r-ip | | | IP address from the outbound server URL |
| r-port | | %p | Port from the outbound server URL |
| r-supplier-dns | | | Hostname of the upstream host (not available for a cache hit) |
| r-supplier-ip | | | IP address used to contact the upstream host (not available for a cache hit) |
| r-supplier-port | | | Port used to contact the upstream host (not available for a cache hit) |
| sc-adapter | proxy.card | | Adapter number of the client's connection to the Appliance |
| sc-connection | | | Unique identifier of the client's connection (i.e. SOCKET) |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-bluecoat-server-connection-socket-errno | server_connection.socket_errno | | Error message associated with a failed attempt to connect to an upstream host |
| s-computername | proxy.name | %N | Configured name of the appliance |
| s-connect-type | | | Upstream connection type (Direct, SOCKS gateway, etc.) |
| s-dns | | | Hostname of the appliance (uses the primary IP address to avoid reverse DNS) |
| s-ip | | %I | IP address of the appliance on which the client established its connection |
| s-port | proxy.port | %P | Port of the appliance on which the client established its connection |
| s-sitename | | %S | The service type used to process the transaction |
| x-service-name | service.name | | The name of the service that handled the transaction |
| x-module-name | module_name | | The SGOS module that is handling the transaction |
| s-supplier-ip | | %D | IP address used to contact the upstream host (not available for a cache hit) |
| s-supplier-name | | %d | Hostname of the upstream host (not available for a cache hit) |
| x-bluecoat-transaction-id | transaction.id | | Unique per-request identifier generated by the appliance (note: this value is not unique across multiple appliances) |
| x-bluecoat-appliance-name | appliance.name | | Configured name of the appliance |
| x-bluecoat-appliance-primary-address | appliance.primary_address | | Primary IP address of the appliance |
| x-bluecoat-proxy-primary-address | proxy.primary_address | | Primary IP address of the appliance |
| x-appliance-serial-number | appliance.serial_number | | The serial number of the appliance |
| x-appliance-mc-certificate-fingerprint | appliance.mc_certificate_finger print | | The fingerprint of the management console certificate |
| x-appliance-product-name | appliance.product_name | | The product name of the appliance -- e.g. Blue Coat SG4xx |
| x-appliance-product-tag | appliance.product_tag | | The product tag of the appliance -- e.g. SG4xx |
| x-appliance-full-version | appliance.full_version | | The full version of the SGOS software |
| x-appliance-first-mac-address | appliance.first_mac_address | | The MAC address of the first installed adapter |
| x-client-address | | | IP address of the client |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-client-connection-bytes | | | Total number of bytes send to and received from the client |
| x-client-ip | | | IP address of the client |
| x-server-connection-bytes | | | Total number of bytes send to and received from the server |
| x-server-adn-connection-bytes | | | Total number of compressed ADN bytes send to and received from the server |
| x-rs-connection-negotiated-cipher | server.connection.negotiated_cipher | | OpenSSL cipher suite negotiated for the client connection |
| x-rs-connection-negotiated-cipher-strength | server.connection.negotiated_cipher.strength | | Strength of the OpenSSL cipher suite negotiated for the server connection |
| x-rs-connection-negotiated-cipher-size | | | Ciphersize of the OpenSSL cipher suite negotiated for the server connection |
| x-rs-connection-negotiated-ssl-version | server.connection.negotiated_ssl_version | | Version of the SSL protocol negotiated for the server connection |
| x-cs-connection-dscp | client.connection.dscp | | DSCP client inbound value |
| x-rs-connection-dscp | server.connection.dscp | | DSCP server inbound value |
| x-sc-connection-dscp-decision | | | DSCP client outbound value |
| x-sr-connection-dscp-decision | | | DSCP server outbound value |
| | | | |
| **Category: dns** | | | |
| x-dns-cs-transport | dns.client_transport | | The transport protocol used by the client connection in a DNS query |
| x-dns-cs-address | dns.request.address | | The address queried in a reverse DNS lookup |
| x-dns-cs-dns | dns.request.name | | The hostname queried in a forward DNS lookup |
| x-dns-cs-opcode | dns.request.opcode | | The DNS OPCODE used in the DNS query |
| x-dns-cs-qtype | dns.request.type | | The DNS QTYPE used in the DNS query |
| x-dns-cs-qclass | dns.request.class | | The DNS QCLASS used in the DNS query |
| x-dns-rs-rcode | dns.response.code | | The DNS RCODE in the response from upstream |
| x-dns-rs-a-records | dns.response.a | | The DNS A RRs in the response from upstream |
| x-dns-rs-cname-records | dns.response.cname | | The DNS CNAME RRs in the response from upstream |
| x-dns-rs-ptr-records | dns.response.ptr | | The DNS PTR RRs in the response from upstream |
| | | | |
| **Category: im** | | | |
| x-im-buddy-id | | | Instant messaging buddy ID |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-im-buddy-name | | | Instant messaging buddy display name |
| x-im-buddy-state | | | Instant messaging buddy state |
| x-im-chat-room-id | | | Instant messaging identifier of the chat room in use |
| x-im-chat-room-members | | | The list of chat room member Ids |
| x-im-chat-room-type | | | The chat room type, one of 'public' or 'public', and possibly 'invite_only', 'voice' and/or 'conference' |
| x-im-client-info | | | The instant messaging client information |
| x-im-user-agent | im.user_agent | | The instant messaging user agent string |
| x-im-file-path | | | Path of the file associated with an instant message |
| x-im-file-size | | | Size of the file associated with an instant message |
| x-im-http-gateway | | | The upstream HTTP gateway used for IM (if any) |
| x-im-message-opcode | im.message.opcode | | The opcode utilized in the instant message |
| x-im-message-reflected | im.message.reflected | | Indicates whether or not the IM message was reflected. |
| x-im-message-route | | | The route of the instance message |
| x-im-message-size | | | Length of the instant message |
| x-im-message-text | | | Text of the instant message |
| x-im-message-type | | | The type of the instant message |
| x-im-method | | | The method associated with the instant message |
| x-im-user-id | | | Instant messaging user identifer |
| x-im-user-name | | | Display name of the client |
| x-im-user-state | | | Instant messaging user state |
| | | | |
| **Category: mapi** | | | |
| x-mapi-method | | | The method associated with the MAPI request |
| x-mapi-user-dn | | | The distinguised name of the user negotiated by MAPI |
| x-mapi-user | | | The name of the user negotiated by MAPI. See x-mapi-user-dn for the fully distinguished name. |
| x-mapi-cs-rpc-count | | | The count of RPC messages received from the client |
| x-mapi-sr-rpc-count | | | The count of RPC messages sent to the server |
| x-mapi-rs-rpc-count | | | The count of RPC messages received from the server |
| x-mapi-sc-rpc-count | | | The count RPC messages sent to the client |

Table 7-6.   Access Log Formats  (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-mapi-endpoint-rpc-count | | | Total number of RPC messages sent to the end point |
| x-mapi-peer-rpc-count | | | Total number of RPC messages sent to the peer |
| | | | |
| **Category: p2p** | | | |
| x-p2p-client-bytes | | | Number of bytes from client |
| x-p2p-client-info | | | The peer-to-peer client information |
| x-p2p-client-type | p2p.client | | The peer-to-peer client type |
| x-p2p-peer-bytes | | | Number of bytes from peer |
| | | | |
| **Category: packets** | | | |
| c-pkts-lost-client | | | Number of packets lost during transmission from server to client and not recovered at the client layer via error correction or at the network layer via UDP resends. |
| c-pkts-lost-cont-net | | | Maximum number of continuously lost packets on the network layer during transmission from server to client |
| c-pkts-lost-net | | | Number of packets lost on the network layer |
| c-pkts-received | | | Number of packets from the server (s-pkts-sent) that are received correctly by the client on the first try |
| c-pkts-recovered-ECC | | | Number of packets repaired and recovered on the client layer |
| c-pkts-recovered-resent | | | Number of packets recovered because they were resent via UDP. |
| c-quality | | | The percentage of packets that were received by the client, indicating the quality of the stream |
| c-resendreqs | | | Number of client requests to receive new packets |
| s-pkts-sent | | | Number of packets from the server |
| | | | |
| **Category: req_rsp_line** | | | |
| cs-method | method | %m | Request method used from client to appliance |
| x-cs-http-method | http.method | | HTTP request method used from client to appliance. Empty for non-HTTP transactions |
| cs-protocol | client.protocol | | Protocol used in the client's request |
| cs-request-line | http.request_line | %r | First line of the client's request |
| x-cs-raw-headers-count | request.raw_headers.count | | Total number of 'raw' headers in the request |
| x-cs-raw-headers-length | request.raw_headers.length | | Total length of 'raw' headers in the request |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs-version | request.version | %V | Protocol and version from the client's request, e.g. HTTP/1.1 |
| x-bluecoat-proxy-via-http-version | proxy.via_http_version | | Default HTTP protocol version of the appliance without protocol decoration (e.g. 1.1 for HTTP/1.1) |
| x-bluecoat-redirect-location | redirect.location | | Redirect location URL specified by a redirect CPL action |
| rs-response-line | | | First line (a.k.a. status line) of the response from an upstream host to the appliance |
| rs-status | response.code | | Protocol status code of the response from an upstream host to the appliance |
| rs-version | response.version | | Protocol and version of the response from an upstream host to the appliance, e.g. HTTP/1.1 |
| sc-status | | %s | Protocol status code from appliance to client |
| x-bluecoat-ssl-failure-reason | ssl_failure_reason | | Upstream SSL negotiation failure reason |
| x-cs-http-version | http.request.version | | HTTP protocol version of request from the client. Does not include protocol qualifier (e.g. 1.1 for HTTP/1.1) |
| x-cs-socks-ip | socks.destination_address | | Destination IP address of a proxied SOCKS request |
| x-cs-socks-port | socks.destination_port | | Destination port of a proxied SOCKS request |
| x-cs-socks-method | socks.method | | Method of a proxied SOCKS request |
| x-cs-socks-version | socks.version | | Version of a proxied SOCKS request. |
| x-cs-socks-compression | | | Used compression in SOCKS client side connection. |
| x-sr-socks-compression | | | Used compression in SOCKS server side connection. |
| x-sc-http-status | http.response.code | | HTTP response code sent from appliance to client |
| x-rs-http-version | http.response.version | | HTTP protocol version of response from the upstream host. Does not include protocol qualifier (e.g. 1.1 for HTTP/1.1) |
| x-sc-http-version | | | HTTP protocol version of response to client. Does not include protocol qualifier (e.g. 1.1 for HTTP/1.1) |
| x-sr-http-version | | | HTTP protocol version of request to the upstream host. Does not include protocol qualifier (e.g. 1.1 for HTTP/1.1) |
| sc(Content-Encoding) | | | Client Response header: Content-Encoding |
| sr(Accept-Encoding) | | | Server Request header: Accept-Encoding |
| | | | |
| **Category: special_token** | | | |
| x-bluecoat-special-amp | amp | | The ampersand character |

Table 7-6.   Access Log Formats  (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-bluecoat-special-apos | apos | | The apostrophe character (a.k.a. single quote) |
| x-bluecoat-special-cr | cr | | Resolves to the carriage return character |
| x-bluecoat-special-crlf | crlf | | Resolves to a carriage return/line feed sequence |
| x-bluecoat-special-empty | empty | %l | Resolves to an empty string |
| x-bluecoat-special-esc | esc | | Resolves to the escape character (ASCII HEX 1B) |
| x-bluecoat-special-gt | gt | | The greater-than character |
| x-bluecoat-special-lf | lf | | The line feed character |
| x-bluecoat-special-lt | lt | | The less-than character |
| x-bluecoat-special-quot | quot | | The double quote character |
| x-bluecoat-special-slash | slash | | The forward slash character |
| | | | |
| **Category: ssl** | | | |
| x-rs-certificate-hostname | server.certificate.hostname | | Hostname from the server's SSL certificate |
| x-rs-certificate-hostname-categories | | | All content categories of the server's SSL certificate's hostname |
| x-rs-certificate-hostname-categories-policy | | | All content categories of the server's SSL certificate's hostname that are defined by CPL. |
| x-rs-certificate-hostname-categories-local | | | All content categories of the server's SSL certificate's hostname that are defined by a Local database. |
| x-rs-certificate-hostname-categories-bluecoat | | | All content categories of the server's SSL certificate's hostname that are defined by Blue Coat Web Filter. |
| x-rs-certificate-hostname-categories-provider | | | All content categories of the server's SSL certificate's hostname that are defined by the current 3rd-party provider. |
| x-rs-certificate-hostname-categories-qualified | | | All content categories of the server's SSL certificate's hostname, qualified by the provider of the category. |
| x-rs-certificate-hostname-category | server.certificate.hostname.category | | Single content category of the server's SSL certificate's hostname |
| x-rs-certificate-valid-from | | | Date from which the certificate presented by the server is valid |
| x-rs-certificate-valid-to | | | Date until which the certificate presented by the server is valid |
| x-rs-certificate-serial-number | | | Serial number of the certificate presented by the server |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-rs-certificate-issuer | | | Issuer of the certificate presented by the server |
| x-rs-certificate-signature-algorithm | | | Signature algorithm in the certificate presented by the server |
| x-rs-certificate-pubkey-algorithm | | | Public key algorithm in the certificate presented by the server |
| x-rs-certificate-version | | | Version of the certificate presented by the server |
| x-rs-certificate-subject | server.certificate.subject | | Subject of the certificate presented by the server |
| x-cs-certificate-common-name | client.certificate.common_name | | Common name in the client certificate |
| x-cs-certificate-valid-from | | | Date from which the certificate presented by the client is valid |
| x-cs-certificate-valid-to | | | Date until which the certificate presented by the client is valid |
| x-cs-certificate-serial-number | | | Serial number of the certificate presented by the client |
| x-cs-certificate-issuer | | | Issuer of the certificate presented by the client |
| x-cs-certificate-signature-algorithm | | | Signature algorithm in the certificate presented by the client |
| x-cs-certificate-pubkey-algorithm | | | Public key algorithm in the certificate presented by the client |
| x-cs-certificate-version | | | Version of the certificate presented by the client |
| x-cs-certificate-subject | client.certificate.subject | | Subject of the certificate presented by the client |
| x-rs-certificate-validate-status | | | Result of validating server SSL certificate |
| x-rs-certificate-observed-errors | | | Errors observed in the server certificate |
| | | | |
| **Category: status** | | | |
| x-bluecoat-release-id | release.id | | The release ID of the ProxySG operating system |
| x-bluecoat-release-version | release.version | | The release version of the ProxySG operating system |
| cs-categories | | | All content categories of the request URL |
| cs-categories-external | | | All content categories of the request URL that are defined by an external service. |
| cs-categories-policy | | | All content categories of the request URL that are defined by CPL. |
| cs-categories-local | | | All content categories of the request URL that are defined by a Local database. |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs-categories-bluecoat | | | All content categories of the request URL that are defined by Blue Coat Web Filter. |
| cs-categories-provider | | | All content categories of the request URL that are defined by the current 3rd-party provider. |
| cs-categories-qualified | | | All content categories of the request URL, qualified by the provider of the category. |
| cs-category | | | Single content category of the request URL (a.k.a. sc-filter-category) |
| cs-uri-categories | | | All content categories of the request URL |
| cs-uri-categories-external | | | All content categories of the request URL that are defined by an external service. |
| cs-uri-categories-policy | | | All content categories of the request URL that are defined by CPL. |
| cs-uri-categories-local | | | All content categories of the request URL that are defined by a Local database. |
| cs-uri-categories-bluecoat | | | All content categories of the request URL that are defined by Blue Coat Web Filter. |
| cs-uri-categories-provider | | | All content categories of the request URL that are defined by the current 3rd-party provider. |
| cs-uri-categories-qualified | | | All content categories of the request URL, qualified by the provider of the category. |
| cs-uri-category | | | Single content category of the request URL (a.k.a. sc-filter-category) |
| x-cs(Referer)-uri-categories | | | All content categories of the Referer header URL |
| x-cs(Referer)-uri-categories-policy | | | All content categories of the Referer header URL that are defined by CPL. |
| x-cs(Referer)-uri-categories-local | | | All content categories of the Referer header URL that are defined by a Local database. |
| x-cs(Referer)-uri-categories-bluecoat | | | All content categories of the Referer header URL that are defined by Blue Coat Web Filter. |
| x-cs(Referer)-uri-categories-provider | | | All content categories of the Referer header URL that are defined by the current 3rd-party provider. |
| x-cs(Referer)-uri-categories-qualified | | | All content categories of the Referer header URL, qualified by the provider of the category. |
| x-cs(Referer)-uri-category | | | Single content category of the Referer header URL (a.k.a. sc-filter-category) |
| r-hierarchy | | | How and where the object was retrieved in the cache hierarchy. |
| sc-filter-category | category | %f | Content filtering category of the request URL |
| sc-filter-result | | %W | Content filtering result: Denied, Proxied or Observed |
| s-action | | %w | What type of action did the Appliance take to process this request. |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| s-cpu-util | | | Average load on the proxy's processor (0%-100%) |
| s-hierarchy | | %H | How and where the object was retrieved in the cache hierarchy. |
| s-icap-info | | %Z | ICAP response information |
| s-icap-status | | %z | ICAP response status |
| x-bluecoat-surfcontrol-category-id | | | The SurfControl specific content category ID. |
| x-bluecoat-surfcontrol-is-denied | | | '1' if the transaction was denied, else '0' |
| x-bluecoat-surfcontrol-is-proxied | | | '0' if transaction is explicitly proxied, '1' if transaction is transparently proxied |
| x-bluecoat-surfcontrol-reporter-id | | | Specialized value for SurfControl reporter |
| x-bluecoat-surfcontrol-reporter-v4 | | | The SurfControl Reporter v4 format |
| x-bluecoat-surfcontrol-reporter-v5 | | | The SurfControl Reporter v5 format |
| x-bluecoat-websense-category-id | | | The Websense specific content category ID |
| x-bluecoat-websense-keyword | | | The Websense specific keyword |
| x-bluecoat-websense-reporter-id | | | The Websense specific reporter category ID |
| x-bluecoat-websense-status | | | The Websense specific numeric status |
| x-bluecoat-websense-user | | | The Websense form of the username |
| x-bluecoat-websense-reporter-protocol-3 | | | The Websense reporter format protocol version 3 |
| x-exception-company-name | exception.company_name | | The company name configured under exceptions |
| x-exception-contact | exception.contact | | Describes who to contact when certain classes of exceptions occur, configured under exceptions (empty if the transaction has not been terminated) |
| x-exception-details | exception.details | | The configurable details of a selecte policy-aware response page (empty if the transaction has not been terminated) |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-exception-header | exception.header | | The header to be associated with an exception response (empty if the transaction has not been terminated) |
| x-exception-help | exception.help | | Help text that accompanies the exception resolved (empty if the transaction has not been terminated) |
| x-exception-id | exception.id | | Identifier of the exception resolved (empty if the transaction has not been terminated) |
| x-exception-last-error | exception.last_error | | The last error recorded for the current transaction. This can provide insight when unexpected problems are occurring (empty if the transaction has not been terminated) |
| x-exception-reason | exception.reason | | Indicates the reason why a particular request was terminated (empty if the transaction has not been terminated) |
| x-exception-sourcefile | exception.sourcefile | | Source filename from which the exception was generated (empty if the transaction has not been terminated) |
| x-exception-sourceline | exception.sourceline | | Source file line number from which the exception was generated (empty if the transaction has not been terminated) |
| x-exception-summary | exception.summary | | Summary of the exception resolved (empty if the transaction has not been terminated) |
| x-exception-category-review-message | exception.category_review_message | | Exception page message that includes a link allowing content categorization to be reviewed and/or disputed. |
| x-exception-category-review-url | exception.category_review_url | | URL where content categorizations can be reviewed and/or disputed. |
| x-patience-javascript | patience_javascript | | Javascript required to allow patience responses |
| x-patience-progress | patience_progress | | The progress of the patience request |
| x-patience-time | patience_time | | The elapsed time of the patience request |
| x-patience-url | patience_url | | The url to be requested for more patience information |
| x-virus-id | icap_virus_id | | Identifier of a virus if one was detected |
| x-virus-details | icap_virus_details | | Details of a virus if one was detected |
| x-icap-error-code | icap_error_code | | ICAP error code |
| x-icap-error-details | icap_error_details | | ICAP error details |
| | | | |
| **Category: streaming** | | | |
| ELFF | CPL | Custom | Description |
| audiocodec | | | Audio codec used in stream. |
| avgbandwidth | | | Average bandwidth (in bits per second) at which the client was connected to the server. |
| channelURL | | | URL to the .nsc file |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| c-buffercount | | | Number of times the client buffered while playing the stream. |
| c-bytes | | | An MMS-only value of the total number of bytes delivered to the client. |
| c-cpu | | | Client computer CPU type. |
| c-hostexe | | | Host application |
| c-hostexever | | | Host application version number |
| c-os | | | Client computer operating system |
| c-osversion | | | Client computer operating system version number |
| c-playerid | | | Globally unique identifier (GUID) of the player |
| c-playerlanguage | | | Client language-country code |
| c-playerversion | | | Version number of the player |
| c-rate | | | Mode of Windows Media Player when the last command event was sent |
| c-starttime | | | Timestamp (in seconds) of the stream when an entry is generated in the log file. |
| c-status | | | Codes that describe client status |
| c-totalbuffertime | | | Time (in seconds) the client used to buffer the stream |
| filelength | | | Length of the file (in seconds). |
| filesize | | | Size of the file (in bytes). |
| protocol | | | Protocol used to access the stream: mms, http, or asfm. |
| s-totalclients | | | Clients connected to the server (but not necessarily receiving streams). |
| transport | | | Transport protocol used (UDP, TCP, multicast, etc.) |
| videocodec | | | Video codec used to encode the stream. |
| x-cache-info | | | Values: UNKNOWN, DEMAND_MISS, DEMAND_PARTIAL_HIT, DEMAND_HIT, LIVE_FROM_ORIGIN, LIVE_PARTIAL_SPLIT, LIVE_SPLIT |
| x-duration | | | Length of time a client played content prior to a client event (FF, REW, Pause, Stop, or jump to marker). |
| x-wm-c-dns | | | Hostname of the client determined from the Windows Media protocol |
| x-wm-c-ip | | | The client IP address determined from the Windows Media protocol |
| x-cs-streaming-client | streaming.client | | Type of streaming client in use (windows_media, real_media, or quicktime). |
| x-rs-streaming-content | streaming.content | | Type of streaming content served. (e.g. windows_media, quicktime) |
| x-streaming-bitrate | bitrate | | The reported client-side bitrate for the stream |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| **Category: time** | | | |
| connect-time | | | Total ms required to connect to the origin server |
| date | date.utc | %x | GMT Date in YYYY-MM-DD format |
| dnslookup-time | | | Total ms cache required to perform the DNS lookup |
| duration | | %T | Time taken (in seconds) to process the request |
| gmttime | | %t | GMT date and time of the user request in format: [DD/MM/YYYY:hh:mm:ss GMT] |
| x-bluecoat-day-utc | day.utc | | GMT/UTC day (as a number) formatted to take up two spaces (e.g. 07 for the 7th of the month) |
| x-bluecoat-hour-utc | hour.utc | | GMT/UTC hour formatted to always take up two spaces (e.g. 01 for 1AM) |
| x-bluecoat-minute-utc | minute.utc | | GMT/UTC minute formatted to always take up two spaces (e.g. 01 for 1 minute past) |
| x-bluecoat-month-utc | month.utc | | GMT/UTC month (as a number) formatted to take up two spaces (e.g. 01 for January) |
| x-bluecoat-monthname-utc | monthname.utc | | GMT/UTC month in the short-form string representation (e.g. Jan for January) |
| x-bluecoat-second-utc | second.utc | | GMT/UTC second formatted to always take up two spaces (e.g. 01 for 1 second past) |
| x-bluecoat-weekday-utc | weekday.utc | | GMT/UTC weekday in the short-form string representation (e.g. Mon for Monday) |
| x-bluecoat-year-utc | year.utc | | GMT/UTC year formatted to always take up four spaces |
| localtime | | %L | Local date and time of the user request in format: [DD/MMM/YYYY:hh:mm:ss +nnnn] |
| x-bluecoat-day | day | | Localtime day (as a number) formatted to take up two spaces (e.g. 07 for the 7th of the month) |
| x-bluecoat-hour | hour | | Localtime hour formatted to always take up two spaces (e.g. 01 for 1AM) |
| x-bluecoat-minute | minute | | Localtime minute formatted to always take up two spaces (e.g. 01 for 1 minute past) |
| x-bluecoat-month | month | | Localtime month (as a number) formatted to take up two spaces (e.g. 01 for January) |
| x-bluecoat-monthname | monthname | | Localtime month in the short-form string representation (e.g. Jan for January) |
| x-bluecoat-second | second | | Localtime second formatted to always take up two spaces (e.g. 01 for 1 second past) |
| x-bluecoat-weekday | weekday | | Localtime weekday in the short-form string representation (e.g. Mon for Monday) |
| x-bluecoat-year | year | | Localtime year formatted to always take up four spaces |
| time | time.utc | %y | GMT time in HH:MM:SS format |
| timestamp | | %g | Unix type timestamp |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| time-taken | | %e | Time taken (in milliseconds) to process the request |
| rs-time-taken | | | Total time taken (in milliseconds) to send the request and receive the response from the origin server |
| x-bluecoat-end-time-wft | | | End local time of the transaction represented as a windows file time |
| x-bluecoat-start-time-wft | | | Start local time of the transaction represented as a windows file time |
| x-bluecoat-end-time-mssql | | | End local time of the transaction represented as a serial date time |
| x-bluecoat-start-time-mssql | | | Start local time of the transaction represented as a serial date time |
| x-cookie-date | cookie_date | | Current date in Cookie time format |
| x-http-date | http_date | | Current date in HTTP time format |
| x-timestamp-unix | | | Seconds since UNIX epoch (Jan 1, 1970) (local time) |
| x-timestamp-unix-utc | | | Seconds since UNIX epoch (Jan 1, 1970) (GMT/UTC) |
| cs-categorization-time-dynamic | | | Time taken (in milliseconds) to dynamically categorize the request URL |
| | | | |
| **Category: url** | | | |
| cs-host | | %v | Hostname from the client's request URL. If URL rewrite policies are used, this field's value is derived from the 'log' URL |
| cs-uri | log_url | %i | The 'log' URL. |
| cs-uri-address | log_url.address | | IP address from the 'log' URL. DNS is used if URL uses a hostname. |
| cs-uri-extension | log_url.extension | | Document extension from the 'log' URL. |
| cs-uri-host | log_url.host | | Hostname from the 'log' URL. |
| cs-uri-hostname | log_url.hostname | | Hostname from the 'log' URL. RDNS is used if the URL uses an IP address. |
| cs-uri-path | log_url.path | %U | Path from the 'log' URL. Does not include query. |
| cs-uri-pathquery | log_url.pathquery | | Path and query from the 'log' URL. |
| cs-uri-port | log_url.port | | Port from the 'log' URL. |
| cs-uri-query | log_url.query | %Q | Query from the 'log' URL. |
| cs-uri-scheme | log_url.scheme | | Scheme from the 'log' URL. |
| cs-uri-stem | | | Stem from the 'log' URL. The stem includes everything up to the end of path, but does not include the query. |
| c-uri | url | | The original URL requested. |
| c-uri-address | url.address | | IP address from the original URL requested. DNS is used if the URL is expressed as a hostname. |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| c-uri-cookie-domain | url.cookie_domain | | The cookie domain of the original URL requested |
| c-uri-extension | url.extension | | Document extension from the original URL requested |
| c-uri-host | url.host | | Hostname from the original URL requested |
| c-uri-hostname | url.hostname | | Hostname from the original URL requested. RDNS is used if the URL is expressed as an IP address |
| c-uri-path | url.path | | Path of the original URL requested without query. |
| c-uri-pathquery | url.pathquery | | Path and query of the original URL requested |
| c-uri-port | url.port | | Port from the original URL requested |
| c-uri-query | url.query | | Query from the original URL requested |
| c-uri-scheme | url.scheme | | Scheme of the original URL requested |
| c-uri-stem | | | Stem of the original URL requested |
| sr-uri | server_url | | URL of the upstream request |
| sr-uri-address | server_url.address | | IP address from the URL used in the upstream request. DNS is used if the URL is expressed as a hostname. |
| sr-uri-extension | server_url.extension | | Document extension from the URL used in the upstream request |
| sr-uri-host | server_url.host | | Hostname from the URL used in the upstream request |
| sr-uri-hostname | server_url.hostname | | Hostname from the URL used in the upstream request. RDNS is used if the URL is expressed as an IP address. |
| sr-uri-path | server_url.path | | Path from the upstream request URL |
| sr-uri-pathquery | server_url.pathquery | | Path and query from the upstream request URL |
| sr-uri-port | server_url.port | | Port from the URL used in the upstream request. |
| sr-uri-query | server_url.query | | Query from the upstream request URL |
| sr-uri-scheme | server_url.scheme | | Scheme from the URL used in the upstream request |
| sr-uri-stem | | | Path from the upstream request URL |
| s-uri | cache_url | | The URL used for cache access |
| s-uri-address | cache_url.address | | IP address from the URL used for cache access. DNS is used if the URL is expressed as a hostname |
| s-uri-extension | cache_url.extension | | Document extension from the URL used for cache access |
| s-uri-host | cache_url.host | | Hostname from the URL used for cache access |
| s-uri-hostname | cache_url.hostname | | Hostname from the URL used for cache access. RDNS is used if the URL uses an IP address |
| s-uri-path | cache_url.path | | Path of the URL used for cache access |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| s-uri-pathquery | cache_url.pathquery | | Path and query of the URL used for cache access |
| s-uri-port | cache_url.port | | Port from the URL used for cache access |
| s-uri-query | cache_url.query | | Query string of the URL used for cache access |
| s-uri-scheme | cache_url.scheme | | Scheme from the URL used for cache access |
| s-uri-stem | | | Stem of the URL used for cache access |
| x-cs(Referer)-uri | request.header.Referer.url | | The URL from the Referer header. |
| x-cs(Referer)-uri-address | request.header.Referer.url.address | | IP address from the 'Referer' URL. DNS is used if URL uses a hostname. |
| x-cs(Referer)-uri-extension | request.header.Referer.url.extension | | Document extension from the 'Referer' URL. |
| x-cs(Referer)-uri-host | request.header.Referer.url.host | | Hostname from the 'Referer' URL. |
| x-cs(Referer)-uri-hostname | request.header.Referer.url.hostname | | Hostname from the 'Referer' URL. RDNS is used if the URL uses an IP address. |
| x-cs(Referer)-uri-path | request.header.Referer.url.path | | Path from the 'Referer' URL. Does not include query. |
| x-cs(Referer)-uri-pathquery | request.header.Referer.url.pathquery | | Path and query from the 'Referer' URL. |
| x-cs(Referer)-uri-port | request.header.Referer.url.port | | Port from the 'Referer' URL. |
| x-cs(Referer)-uri-query | request.header.Referer.url.query | | Query from the 'Referer' URL. |
| x-cs(Referer)-uri-scheme | request.header.Referer.url.scheme | | Scheme from the 'Referer' URL. |
| x-cs(Referer)-uri-stem | | | Stem from the 'Referer' URL. The stem includes everything up to the end of path, but does not include the query. |
| x-cs-raw-uri | raw_url | | The 'raw' request URL. |
| x-cs-raw-uri-host | raw_url.host | | Hostname from the 'raw' URL. |
| x-cs-raw-uri-port | raw_url.port | | Port string from the 'raw' URL. |
| x-cs-raw-uri-scheme | raw_url.scheme | | Scheme string from the 'raw' URL. |
| x-cs-raw-uri-path | raw_url.path | | Path from the 'raw' request URL. Does not include query. |
| x-cs-raw-uri-pathquery | raw_url.pathquery | | Path and query from the 'raw' request URL. |
| x-cs-raw-uri-query | raw_url.query | | Query from the 'raw' request URL. |
| x-cs-raw-uri-stem | | | Stem from the 'raw' request URL. The stem includes everything up to the end of path, but does not include the query. |
| | | | |
| **Category: user** | | | |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs-auth-group | group | | One group that an authenticated user belongs to. If a user belongs to multiple groups, the group logged is determined by the Group Log Order configuration specified in VPM. If Group Log Order is not specified, an arbitrary group is logged. Note that only groups referenced by policy are considered. |
| cs-auth-groups | groups | | List of groups that an authenticated user belongs to. Note that only groups referenced by policy are included. |
| cs-auth-type | | | Client-side: authentication type (basic, ntlm, etc.) |
| cs-realm | realm | | Authentication realm that the user was challenged in. |
| cs-user | | %u | Qualified username for NTLM. Relative username for other protocols |
| cs-userdn | user | | Full username of a client authenticated to the proxy (fully distinguished) |
| x-cs-user-authorization-name | user.authorization_name | | Username used to authorize a client authenticated to the proxy |
| cs-username | user.name | | Relative username of a client authenticated to the proxy (i.e. not fully distinguished) |
| sc-auth-status | | | Client-side: Authorization status |
| x-agent-sso-cookie | | | The authentication agent single signon cookie |
| x-cache-user | | | Relative username of a client authenticated to the proxy (i.e. not fully distinguished) (same as cs-username) |
| x-cs-auth-domain | user.domain | | The domain of the authenticated user. |
| x-cs-auth-form-action-url | | | The URL to submit the authentication form to. |
| x-cs-auth-form-domain-field | | | The authentication form input field for the user's domain. |
| x-cs-auth-request-id | | | The bas64 encoded string containing the original request information during forms based authentication |
| x-cs-username-or-ip | | | Used to identify the user using either their authenticated proxy username or, if that is unavailable, their IP address. |
| x-radius-splash-session-id | | | Session ID made available through RADIUS when configured for session management |
| x-radius-splash-username | | | Username made available through RADIUS when configured for session management |
| x-user-x509-issuer | user.x509.issuer | | If the user was authenticated via an X.509 certificate, this is the issuer of the certificate as an RFC2253 DN |
| x-user-x509-serial-number | user.x509.serialNumber | | If the user was authenticated via an X.509 certificate, this is the serial number from the certificate as a hexadecimal number. |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| x-user-x509-subject | user.x509.subject | | If the user was authenticated via an X.509 certificate, this is the subject of the certificate as an RFC2253 DN |
| x-auth-challenge-string | | | The authentication challenge to display to the user. |
| x-auth-private-challenge-state | | | The private state required to manage an authentication challenge |
| | | | |
| **Category: ci_request_header** | | | |
| cs(Accept) | request.header.Accept | | Request header: Accept |
| cs(Accept)-length | request.header.Accept.length | | Length of HTTP request header: Accept |
| cs(Accept)-count | request.header.Accept.count | | Number of HTTP request header: Accept |
| cs(Accept-Charset) | request.header.Accept-Charset | | Request header: Accept-Charset |
| cs(Accept-Charset)-length | request.header.Accept-Charset.length | | Length of HTTP request header: Accept-Charset |
| cs(Accept-Charset)-count | request.header.Accept-Charset.count | | Number of HTTP request header: Accept-Charset |
| cs(Accept-Encoding) | request.header.Accept-Encoding | | Request header: Accept-Encoding |
| cs(Accept-Encoding)-length | request.header.Accept-Encoding.length | | Length of HTTP request header: Accept-Encoding |
| cs(Accept-Encoding)-count | request.header.Accept-Encoding.count | | Number of HTTP request header: Accept-Encoding |
| cs(Accept-Language) | request.header.Accept-Language | | Request header: Accept-Language |
| cs(Accept-Language)-length | request.header.Accept-Language.length | | Length of HTTP request header: Accept-Language |
| cs(Accept-Language)-count | request.header.Accept-Language.count | | Number of HTTP request header: Accept-Language |
| cs(Accept-Ranges) | request.header.Accept-Ranges | | Request header: Accept-Ranges |
| cs(Accept-Ranges)-length | request.header.Accept-Ranges.length | | Length of HTTP request header: Accept-Ranges |
| cs(Accept-Ranges)-count | request.header.Accept-Ranges.count | | Number of HTTP request header: Accept-Ranges |
| cs(Age) | request.header.Age | | Request header: Age |
| cs(Age)-length | request.header.Age.length | | Length of HTTP request header: Age |
| cs(Age)-count | request.header.Age.count | | Number of HTTP request header: Age |
| cs(Allow) | request.header.Allow | | Request header: Allow |
| cs(Allow)-length | request.header.Allow.length | | Length of HTTP request header: Allow |
| cs(Allow)-count | request.header.Allow.count | | Number of HTTP request header: Allow |
| cs(Authentication-Info) | request.header.Authentication-Info | | Request header: Authentication-Info |
| cs(Authentication-Info)-length | request.header.Authentication-Info.length | | Length of HTTP request header: Authentication-Info |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs(Authentication-Info)-count | request.header.Authentication-Info.count | | Number of HTTP request header: Authentication-Info |
| cs(Authorization) | request.header.Authorization | | Request header: Authorization |
| cs(Authorization)-length | request.header.Authorization.length | | Length of HTTP request header: Authorization |
| cs(Authorization)-count | request.header.Authorization.count | | Number of HTTP request header: Authorization |
| cs(Cache-Control) | request.header.Cache-Control | | Request header: Cache-Control |
| cs(Cache-Control)-length | request.header.Cache-Control.length | | Length of HTTP request header: Cache-Control |
| cs(Cache-Control)-count | request.header.Cache-Control.count | | Number of HTTP request header: Cache-Control |
| cs(Client-IP) | request.header.Client-IP | | Request header: Client-IP |
| cs(Client-IP)-length | request.header.Client-IP.length | | Length of HTTP request header: Client-IP |
| cs(Client-IP)-count | request.header.Client-IP.count | | Number of HTTP request header: Client-IP |
| cs(Connection) | request.header.Connection | | Request header: Connection |
| cs(Connection)-length | request.header.Connection.length | | Length of HTTP request header: Connection |
| cs(Connection)-count | request.header.Connection.count | | Number of HTTP request header: Connection |
| cs(Content-Disposition) | request.header.Content-Disposition | | Request header: Content-Disposition |
| cs(Content-Disposition)-length | request.header.Content-Disposition.length | | Length of HTTP request header: Content-Disposition |
| cs(Content-Disposition)-count | request.header.Content-Disposition.count | | Number of HTTP request header: Content-Disposition |
| cs(Content-Encoding) | request.header.Content-Encoding | | Request header: Content-Encoding |
| cs(Content-Encoding)-length | request.header.Content-Encoding.length | | Length of HTTP request header: Content-Encoding |
| cs(Content-Encoding)-count | request.header.Content-Encoding.count | | Number of HTTP request header: Content-Encoding |
| cs(Content-Language) | request.header.Content-Language | | Request header: Content-Language |
| cs(Content-Language)-length | request.header.Content-Language.length | | Length of HTTP request header: Content-Language |
| cs(Content-Language)-count | request.header.Content-Language.count | | Number of HTTP request header: Content-Language |
| cs(Content-Length) | request.header.Content-Length | | Request header: Content-Length |
| cs(Content-Length)-length | request.header.Content-Length.length | | Length of HTTP request header: Content-Length |
| cs(Content-Length)-count | request.header.Content-Length.count | | Number of HTTP request header: Content-Length |
| cs(Content-Location) | request.header.Content-Location | | Request header: Content-Location |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs(Content-Location)-length | request.header.Content-Location.length | | Length of HTTP request header: Content-Location |
| cs(Content-Location)-count | request.header.Content-Location.count | | Number of HTTP request header: Content-Location |
| cs(Content-MD5) | request.header.Content-MD5 | | Request header: Content-MD5 |
| cs(Content-MD5)-length | request.header.Content-MD5.length | | Length of HTTP request header: Content-MD5 |
| cs(Content-MD5)-count | request.header.Content-MD5.count | | Number of HTTP request header: Content-MD5 |
| cs(Content-Range) | request.header.Content-Range | | Request header: Content-Range |
| cs(Content-Range)-length | request.header.Content-Range.length | | Length of HTTP request header: Content-Range |
| cs(Content-Range)-count | request.header.Content-Range.count | | Number of HTTP request header: Content-Range |
| cs(Content-Type) | request.header.Content-Type | | Request header: Content-Type |
| cs(Content-Type)-length | request.header.Content-Type.length | | Length of HTTP request header: Content-Type |
| cs(Content-Type)-count | request.header.Content-Type.count | | Number of HTTP request header: Content-Type |
| cs(Cookie) | request.header.Cookie | %C | Request header: Cookie |
| cs(Cookie)-length | request.header.Cookie.length | | Length of HTTP request header: Cookie |
| cs(Cookie)-count | request.header.Cookie.count | | Number of HTTP request header: Cookie |
| cs(Cookie2) | request.header.Cookie2 | | Request header: Cookie2 |
| cs(Cookie2)-length | request.header.Cookie2.length | | Length of HTTP request header: Cookie2 |
| cs(Cookie2)-count | request.header.Cookie2.count | | Number of HTTP request header: Cookie2 |
| cs(Date) | request.header.Date | | Request header: Date |
| cs(Date)-length | request.header.Date.length | | Length of HTTP request header: Date |
| cs(Date)-count | request.header.Date.count | | Number of HTTP request header: Date |
| cs(Etag) | request.header.Etag | | Request header: Etag |
| cs(Etag)-length | request.header.Etag.length | | Length of HTTP request header: Etag |
| cs(Etag)-count | request.header.Etag.count | | Number of HTTP request header: Etag |
| cs(Expect) | request.header.Expect | | Request header: Expect |
| cs(Expect)-length | request.header.Expect.length | | Length of HTTP request header: Expect |
| cs(Expect)-count | request.header.Expect.count | | Number of HTTP request header: Expect |
| cs(Expires) | request.header.Expires | | Request header: Expires |
| cs(Expires)-length | request.header.Expires.length | | Length of HTTP request header: Expires |
| cs(Expires)-count | request.header.Expires.count | | Number of HTTP request header: Expires |
| cs(From) | request.header.From | | Request header: From |
| cs(From)-length | request.header.From.length | | Length of HTTP request header: From |
| cs(From)-count | request.header.From.count | | Number of HTTP request header: From |
| cs(Front-End-HTTPS) | request.header.Front-End-HTTPS | | Request header: Front-End-HTTPS |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs(Front-End-HTTPS)-length | request.header.Front-End-HTTPS.length | | Length of HTTP request header: Front-End-HTTPS |
| cs(Front-End-HTTPS)-count | request.header.Front-End-HTTPS.count | | Number of HTTP request header: Front-End-HTTPS |
| cs(Host) | request.header.Host | | Request header: Host |
| cs(Host)-length | request.header.Host.length | | Length of HTTP request header: Host |
| cs(Host)-count | request.header.Host.count | | Number of HTTP request header: Host |
| cs(If-Match) | request.header.If-Match | | Request header: If-Match |
| cs(If-Match)-length | request.header.If-Match.length | | Length of HTTP request header: If-Match |
| cs(If-Match)-count | request.header.If-Match.count | | Number of HTTP request header: If-Match |
| cs(If-Modified-Since) | request.header.If-Modified-Since | | Request header: If-Modified-Since |
| cs(If-Modified-Since)-length | request.header.If-Modified-Since.length | | Length of HTTP request header: If-Modified-Since |
| cs(If-Modified-Since)-count | request.header.If-Modified-Since.count | | Number of HTTP request header: If-Modified-Since |
| cs(If-None-Match) | request.header.If-None-Match | | Request header: If-None-Match |
| cs(If-None-Match)-length | request.header.If-None-Match.length | | Length of HTTP request header: If-None-Match |
| cs(If-None-Match)-count | request.header.If-None-Match.count | | Number of HTTP request header: If-None-Match |
| cs(If-Range) | request.header.If-Range | | Request header: If-Range |
| cs(If-Range)-length | request.header.If-Range.length | | Length of HTTP request header: If-Range |
| cs(If-Range)-count | request.header.If-Range.count | | Number of HTTP request header: If-Range |
| cs(If-Unmodified-Since) | request.header.If-Unmodified-Since | | Request header: If-Unmodified-Since |
| cs(If-Unmodified-Since)-length | request.header.If-Unmodified-Since.length | | Length of HTTP request header: If-Unmodified-Since |
| cs(If-Unmodified-Since)-count | request.header.If-Unmodified-Since.count | | Number of HTTP request header: If-Unmodified-Since |
| cs(Last-Modified) | request.header.Last-Modified | | Request header: Last-Modified |
| cs(Last-Modified)-length | request.header.Last-Modified.length | | Length of HTTP request header: Last-Modified |
| cs(Last-Modified)-count | request.header.Last-Modified.count | | Number of HTTP request header: Last-Modified |
| cs(Location) | request.header.Location | | Request header: Location |
| cs(Location)-length | request.header.Location.length | | Length of HTTP request header: Location |
| cs(Location)-count | request.header.Location.count | | Number of HTTP request header: Location |
| cs(Max-Forwards) | request.header.Max-Forwards | | Request header: Max-Forwards |
| cs(Max-Forwards)-length | request.header.Max-Forwards.length | | Length of HTTP request header: Max-Forwards |
| cs(Max-Forwards)-count | request.header.Max-Forwards.count | | Number of HTTP request header: Max-Forwards |
| cs(Meter) | request.header.Meter | | Request header: Meter |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs(Meter)-length | request.header.Meter.length | | Length of HTTP request header: Meter |
| cs(Meter)-count | request.header.Meter.count | | Number of HTTP request header: Meter |
| cs(P3P) | request.header.P3P | | Request header: P3P |
| cs(P3P)-length | request.header.P3P.length | | Length of HTTP request header: P3P |
| cs(P3P)-count | request.header.P3P.count | | Number of HTTP request header: P3P |
| cs(Pragma) | request.header.Pragma | | Request header: Pragma |
| cs(Pragma)-length | request.header.Pragma.length | | Length of HTTP request header: Pragma |
| cs(Pragma)-count | request.header.Pragma.count | | Number of HTTP request header: Pragma |
| cs(Proxy-Authenticate) | request.header.Proxy-Authenticate | | Request header: Proxy-Authenticate |
| cs(Proxy-Authenticate)-length | request.header.Proxy-Authenticate.length | | Length of HTTP request header: Proxy-Authenticate |
| cs(Proxy-Authenticate)-count | request.header.Proxy-Authenticate.count | | Number of HTTP request header: Proxy-Authenticate |
| cs(Proxy-Authorization) | request.header.Proxy-Authorization | | Request header: Proxy-Authorization |
| cs(Proxy-Authorization)-length | request.header.Proxy-Authorization.length | | Length of HTTP request header: Proxy-Authorization |
| cs(Proxy-Authorization)-count | request.header.Proxy-Authorization.count | | Number of HTTP request header: Proxy-Authorization |
| cs(Proxy-Connection) | request.header.Proxy-Connection | | Request header: Proxy-Connection |
| cs(Proxy-Connection)-length | request.header.Proxy-Connection.length | | Length of HTTP request header: Proxy-Connection |
| cs(Proxy-Connection)-count | request.header.Proxy-Connection.count | | Number of HTTP request header: Proxy-Connection |
| cs(Range) | request.header.Range | | Request header: Range |
| cs(Range)-length | request.header.Range.length | | Length of HTTP request header: Range |
| cs(Range)-count | request.header.Range.count | | Number of HTTP request header: Range |
| cs(Referer) | request.header.Referer | %R | Request header: Referer |
| cs(Referer)-length | request.header.Referer.length | | Length of HTTP request header: Referer |
| cs(Referer)-count | request.header.Referer.count | | Number of HTTP request header: Referer |
| cs(Refresh) | request.header.Refresh | | Request header: Refresh |
| cs(Refresh)-length | request.header.Refresh.length | | Length of HTTP request header: Refresh |
| cs(Refresh)-count | request.header.Refresh.count | | Number of HTTP request header: Refresh |
| cs(Retry-After) | request.header.Retry-After | | Request header: Retry-After |
| cs(Retry-After)-length | request.header.Retry-After.length | | Length of HTTP request header: Retry-After |
| cs(Retry-After)-count | request.header.Retry-After.count | | Number of HTTP request header: Retry-After |
| cs(Server) | request.header.Server | | Request header: Server |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs(Server)-length | request.header.Server.length | | Length of HTTP request header: Server |
| cs(Server)-count | request.header.Server.count | | Number of HTTP request header: Server |
| cs(Set-Cookie) | request.header.Set-Cookie | | Request header: Set-Cookie |
| cs(Set-Cookie)-length | request.header.Set-Cookie.length | | Length of HTTP request header: Set-Cookie |
| cs(Set-Cookie)-count | request.header.Set-Cookie.count | | Number of HTTP request header: Set-Cookie |
| cs(Set-Cookie2) | request.header.Set-Cookie2 | | Request header: Set-Cookie2 |
| cs(Set-Cookie2)-length | request.header.Set-Cookie2.length | | Length of HTTP request header: Set-Cookie2 |
| cs(Set-Cookie2)-count | request.header.Set-Cookie2.count | | Number of HTTP request header: Set-Cookie2 |
| cs(TE) | request.header.TE | | Request header: TE |
| cs(TE)-length | request.header.TE.length | | Length of HTTP request header: TE |
| cs(TE)-count | request.header.TE.count | | Number of HTTP request header: TE |
| cs(Trailer) | request.header.Trailer | | Request header: Trailer |
| cs(Trailer)-length | request.header.Trailer.length | | Length of HTTP request header: Trailer |
| cs(Trailer)-count | request.header.Trailer.count | | Number of HTTP request header: Trailer |
| cs(Transfer-Encoding) | request.header.Transfer-Encoding | | Request header: Transfer-Encoding |
| cs(Transfer-Encoding)-length | request.header.Transfer-Encoding.length | | Length of HTTP request header: Transfer-Encoding |
| cs(Transfer-Encoding)-count | request.header.Transfer-Encoding.count | | Number of HTTP request header: Transfer-Encoding |
| cs(Upgrade) | request.header.Upgrade | | Request header: Upgrade |
| cs(Upgrade)-length | request.header.Upgrade.length | | Length of HTTP request header: Upgrade |
| cs(Upgrade)-count | request.header.Upgrade.count | | Number of HTTP request header: Upgrade |
| cs(User-Agent) | request.header.User-Agent | %A | Request header: User-Agent |
| cs(User-Agent)-length | request.header.User-Agent.length | | Length of HTTP request header: User-Agent |
| cs(User-Agent)-count | request.header.User-Agent.count | | Number of HTTP request header: User-Agent |
| cs(Vary) | request.header.Vary | | Request header: Vary |
| cs(Vary)-length | request.header.Vary.length | | Length of HTTP request header: Vary |
| cs(Vary)-count | request.header.Vary.count | | Number of HTTP request header: Vary |
| cs(Via) | request.header.Via | | Request header: Via |
| cs(Via)-length | request.header.Via.length | | Length of HTTP request header: Via |
| cs(Via)-count | request.header.Via.count | | Number of HTTP request header: Via |
| cs(WWW-Authenticate) | request.header.WWW-Authenticate | | Request header: WWW-Authenticate |
| cs(WWW-Authenticate)-length | request.header.WWW-Authenticate.length | | Length of HTTP request header: WWW-Authenticate |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| cs(WWW-Authenticate)-count | request.header.WWW-Authenticate.count | | Number of HTTP request header: WWW-Authenticate |
| cs(Warning) | request.header.Warning | | Request header: Warning |
| cs(Warning)-length | request.header.Warning.length | | Length of HTTP request header: Warning |
| cs(Warning)-count | request.header.Warning.count | | Number of HTTP request header: Warning |
| cs(X-BlueCoat-Error) | request.header.X-BlueCoat-Error | | Request header: X-BlueCoat-Error |
| cs(X-BlueCoat-Error)-length | request.header.X-BlueCoat-Error.length | | Length of HTTP request header: X-BlueCoat-Error |
| cs(X-BlueCoat-Error)-count | request.header.X-BlueCoat-Error.count | | Number of HTTP request header: X-BlueCoat-Error |
| cs(X-BlueCoat-MC-Client-Ip) | request.header.X-BlueCoat-MC-Client-Ip | | Request header: X-BlueCoat-MC-Client-Ip |
| cs(X-BlueCoat-MC-Client-Ip)-length | request.header.X-BlueCoat-MC-Client-Ip.length | | Length of HTTP request header: X-BlueCoat-MC-Client-Ip |
| cs(X-BlueCoat-MC-Client-Ip)-count | request.header.X-BlueCoat-MC-Client-Ip.count | | Number of HTTP request header: X-BlueCoat-MC-Client-Ip |
| cs(X-BlueCoat-Via) | request.header.X-BlueCoat-Via | | Request header: X-BlueCoat-Via |
| cs(X-BlueCoat-Via)-length | request.header.X-BlueCoat-Via.length | | Length of HTTP request header: X-BlueCoat-Via |
| cs(X-BlueCoat-Via)-count | request.header.X-BlueCoat-Via.count | | Number of HTTP request header: X-BlueCoat-Via |
| cs(X-Forwarded-For) | request.header.X-Forwarded-For | %X | Request header: X-Forwarded-For |
| cs(X-Forwarded-For)-length | request.header.X-Forwarded-For.length | | Length of HTTP request header: X-Forwarded-For |
| cs(X-Forwarded-For)-count | request.header.X-Forwarded-For.count | | Number of HTTP request header: X-Forwarded-For |
| | | | |
| **Category: si_response_header** | | | |
| rs(Accept) | response.header.Accept | | Response header: Accept |
| rs(Accept-Charset) | response.header.Accept-Charset | | Response header: Accept-Charset |
| rs(Accept-Encoding) | response.header.Accept-Encoding | | Response header: Accept-Encoding |
| rs(Accept-Language) | response.header.Accept-Language | | Response header: Accept-Language |
| rs(Accept-Ranges) | response.header.Accept-Ranges | | Response header: Accept-Ranges |
| rs(Age) | response.header.Age | | Response header: Age |
| rs(Allow) | response.header.Allow | | Response header: Allow |
| rs(Authentication-Info) | response.header.Authentication-Info | | Response header: Authentication-Info |
| rs(Authorization) | response.header.Authorization | | Response header: Authorization |
| rs(Cache-Control) | response.header.Cache-Control | | Response header: Cache-Control |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| rs(Client-IP) | response.header.Client-IP | | Response header: Client-IP |
| rs(Connection) | response.header.Connection | | Response header: Connection |
| rs(Content-Disposition) | response.header.Content-Disposition | | Response header: Content-Disposition |
| rs(Content-Encoding) | response.header.Content-Encoding | | Response header: Content-Encoding |
| rs(Content-Language) | response.header.Content-Language | | Response header: Content-Language |
| rs(Content-Length) | response.header.Content-Length | | Response header: Content-Length |
| rs(Content-Location) | response.header.Content-Location | | Response header: Content-Location |
| rs(Content-MD5) | response.header.Content-MD5 | | Response header: Content-MD5 |
| rs(Content-Range) | response.header.Content-Range | | Response header: Content-Range |
| rs(Content-Type) | response.header.Content-Type | %c | Response header: Content-Type |
| rs(Cookie) | response.header.Cookie | | Response header: Cookie |
| rs(Cookie2) | response.header.Cookie2 | | Response header: Cookie2 |
| rs(Date) | response.header.Date | | Response header: Date |
| rs(Etag) | response.header.Etag | | Response header: Etag |
| rs(Expect) | response.header.Expect | | Response header: Expect |
| rs(Expires) | response.header.Expires | | Response header: Expires |
| rs(From) | response.header.From | | Response header: From |
| rs(Front-End-HTTPS) | response.header.Front-End-HTTPS | | Response header: Front-End-HTTPS |
| rs(Host) | response.header.Host | | Response header: Host |
| rs(If-Match) | response.header.If-Match | | Response header: If-Match |
| rs(If-Modified-Since) | response.header.If-Modified-Since | | Response header: If-Modified-Since |
| rs(If-None-Match) | response.header.If-None-Match | | Response header: If-None-Match |
| rs(If-Range) | response.header.If-Range | | Response header: If-Range |
| rs(If-Unmodified-Since) | response.header.If-Unmodified-Since | | Response header: If-Unmodified-Since |
| rs(Last-Modified) | response.header.Last-Modified | | Response header: Last-Modified |
| rs(Location) | response.header.Location | | Response header: Location |
| rs(Max-Forwards) | response.header.Max-Forwards | | Response header: Max-Forwards |
| rs(Meter) | response.header.Meter | | Response header: Meter |
| rs(P3P) | response.header.P3P | | Response header: P3P |
| rs(Pragma) | response.header.Pragma | | Response header: Pragma |
| rs(Proxy-Authenticate) | response.header.Proxy-Authenticate | | Response header: Proxy-Authenticate |
| rs(Proxy-Authorization) | response.header.Proxy-Authorization | | Response header: Proxy-Authorization |

Table 7-6. Access Log Formats (Continued)

| ELFF | CPL | Custom | Description |
|---|---|---|---|
| rs(Proxy-Connection) | response.header.Proxy-Connection | | Response header: Proxy-Connection |
| rs(Range) | response.header.Range | | Response header: Range |
| rs(Referer) | response.header.Referer | | Response header: Referer |
| rs(Refresh) | response.header.Refresh | | Response header: Refresh |
| rs(Retry-After) | response.header.Retry-After | | Response header: Retry-After |
| rs(Server) | response.header.Server | | Response header: Server |
| rs(Set-Cookie) | response.header.Set-Cookie | | Response header: Set-Cookie |
| rs(Set-Cookie2) | response.header.Set-Cookie2 | | Response header: Set-Cookie2 |
| rs(TE) | response.header.TE | | Response header: TE |
| rs(Trailer) | response.header.Trailer | | Response header: Trailer |
| rs(Transfer-Encoding) | response.header.Transfer-Encoding | | Response header: Transfer-Encoding |
| rs(Upgrade) | response.header.Upgrade | | Response header: Upgrade |
| rs(User-Agent) | response.header.User-Agent | | Response header: User-Agent |
| rs(Vary) | response.header.Vary | | Response header: Vary |
| rs(Via) | response.header.Via | | Response header: Via |
| rs(WWW-Authenticate) | response.header.WWW-Authenticate | | Response header: WWW-Authenticate |
| rs(Warning) | response.header.Warning | | Response header: Warning |
| rs(X-BlueCoat-Error) | response.header.X-BlueCoat-Error | | Response header: X-BlueCoat-Error |
| rs(X-BlueCoat-MC-Client-Ip) | response.header.X-BlueCoat-MC-Client-Ip | | Response header: X-BlueCoat-MC-Client-Ip |
| rs(X-BlueCoat-Via) | response.header.X-BlueCoat-Via | | Response header: X-BlueCoat-Via |
| rs(X-Forwarded-For) | response.header.X-Forwarded-For | | Response header: X-Forwarded-For |

# Index