Apache HTTP Server Version 2.4

Apache Module mod_log_config

Description:	Logging of the requests made to the server
Status:	Base
Module Identifier:	log_config_module
Source File:	mod_log_config.c

Summary

This module provides for flexible logging of client requests. Logs are written in a customizable format, and may be written directly to a file, or to an external program. Conditional logging is provided so that individual requests may be included or excluded from the logs based on characteristics of the request.

Three directives are provided by this module: `TransferLog` to create a log file, `LogFormat` to set a custom format, and `CustomLog` to define a log file and format in one step. The `TransferLog` and `CustomLog` directives can be used multiple times in each server to cause each request to be logged to multiple files.

Custom Log Formats

The format argument to the `LogFormat` and `CustomLog` directives is a string. This string is used to log each request to the log file. It can contain literal characters copied into the log files and the C-style control characters "\n" and "\t" to represent new-lines and tabs. Literal quotes and backslashes should be escaped with backslashes.

The characteristics of the request itself are logged by placing "%" directives in the format string, which are replaced in the log file by the values as follows:

Format String	Description
%%	The percent sign.
%a	Client IP address of the request (see the `mod_remoteip` module).
%{c}a	Underlying peer IP address of the connection (see the `mod_remoteip` module).
%A	Local IP-address.
%B	Size of response in bytes, excluding HTTP headers.
%b	Size of response in bytes, excluding HTTP headers. In CLF format, *i.e.* a '-' rather than a 0 when no bytes are sent.
%{*VARNAME*}C	The contents of cookie *VARNAME* in the request sent to the server. Only version 0 cookies are fully supported.
%D	The time taken to serve the request, in microseconds.
%{*VARNAME*}e	The contents of the environment variable *VARNAME*.
%f	Filename.
%h	Remote hostname. Will log the IP address if `HostnameLookups` is set to `Off`, which is the default. If it logs the hostname for only a few hosts, you probably have access control directives mentioning them by name. See the Require host documentation.
%{c}h	Like %h, but always reports on the hostname of the underlying TCP connection and not any modifications to the remote hostname by modules like `mod_remoteip`.
%H	The request protocol.
%{*VARNAME*}i	The contents of *VARNAME*: header line(s) in the request sent to the server. Changes made by other modules (e.g. `mod_headers`) affect this. If you're interested in what the request header was prior to when most modules would have modified it, use `mod_setenvif` to copy the header into an internal environment variable and log that value with the %{*VARNAME*}e described above.
%k	Number of keepalive requests handled on this connection. Interesting if `KeepAlive` is being used, so that, for example, a '1' means the first keepalive request after the initial one, '2' the second, etc...; otherwise this is always 0 (indicating the initial request).
%l	Remote logname (from identd, if supplied). This will return a dash unless `mod_ident` is present and `IdentityCheck` is set `On`.
%L	The request log ID from the error log (or '-' if nothing has been logged to the error log for this request). Look for the matching error log line to see what request caused what error.
%m	The request method.
%{*VARNAME*}n	The contents of note *VARNAME* from another module.
%{*VARNAME*}o	The contents of *VARNAME*: header line(s) in the reply.
%p	The canonical port of the server serving the request.
%{*format*}p	The canonical port of the server serving the request, or the server's actual port, or the client's actual port. Valid formats are `canonical`, `local`, or `remote`.
%P	The process ID of the child that serviced the request.
%{*format*}P	The process ID or thread ID of the child that serviced the request. Valid formats are `pid`, `tid`, and `hextid`.
%q	The query string (prepended with a ? if a query string exists, otherwise an empty string).
%r	First line of request.
%R	The handler generating the response (if any).
%s	Status. For requests that have been internally redirected, this is the status of the *original* request. Use %>s for the final status.
%t	Time the request was received, in the format `[18/Sep/2011:19:18:28 -0400]`. The last number indicates the timezone offset from GMT
%{*format*}t	The time, in the form given by format, which should be in an extended `strftime(3)` format (potentially localized). If the format starts with `begin:` (default) the time is taken at the beginning of the request processing. If it starts with `end:` it is the time when the log entry gets written, close to the end of the request processing. In addition to the formats supported by `strftime(3)`, the following format tokens are supported: `sec` number of seconds since the Epoch `msec` number of milliseconds since the Epoch `usec` number of microseconds since the Epoch `msec_frac` millisecond fraction `usec_frac` microsecond fraction These tokens can not be combined with each other or `strftime(3)` formatting in the same format string. You can use multiple %{*format*}t tokens instead.
%T	The time taken to serve the request, in seconds.
%{*UNIT*}T	The time taken to serve the request, in a time unit given by `UNIT`. Valid units are `ms` for milliseconds, `us` for microseconds, and `s` for seconds. Using `s` gives the same result as %T without any format; using `us` gives the same result as %D. Combining %T with a unit is available in 2.4.13 and later.
%u	Remote user if the request was authenticated. May be bogus if return status (%s) is 401 (unauthorized).
%U	The URL path requested, not including any query string.
%v	The canonical `ServerName` of the server serving the request.

%V	The server name according to the `UseCanonicalName` setting.
%X	Connection status when response is completed: X = Connection aborted before the response completed. + = Connection may be kept alive after the response is sent. - = Connection will be closed after the response is sent.
%I	Bytes received, including request and headers. Cannot be zero. You need to enable `mod_logio` to use this.
%O	Bytes sent, including headers. May be zero in rare cases such as when a request is aborted before a response is sent. You need to enable `mod_logio` to use this.
%S	Bytes transferred (received and sent), including request and headers, cannot be zero. This is the combination of %I and %O. You need to enable `mod_logio` to use this.
%{*VARNAME*}^ti	The contents of *VARNAME*: trailer line(s) in the request sent to the server.
%{*VARNAME*}^to	The contents of *VARNAME*: trailer line(s) in the response sent from the server.

Modifiers

Particular items can be restricted to print only for responses with specific HTTP status codes by placing a comma-separated list of status codes immediately following the "%". The status code list may be preceded by a "!" to indicate negation.

Format String	Meaning
%400,501{User-agent}i	Logs `User-agent` on 400 errors and 501 errors only. For other status codes, the literal string "-" will be logged.
%!200,304,302{Referer}i	Logs `Referer` on all requests that do *not* return one of the three specified codes, "-" otherwise.

The modifiers "<" and ">" can be used for requests that have been internally redirected to choose whether the original or final (respectively) request should be consulted. By default, the % directives %s, %U, %T, %D, and %r look at the original request while all others look at the final request. So for example, %>s can be used to record the final status of the request and %<u can be used to record the original authenticated user on a request that is internally redirected to an unauthenticated resource.

Format Notes

For security reasons, starting with version 2.0.46, non-printable and other special characters in %r, %i and %o are escaped using \x*hh* sequences, where *hh* stands for the hexadecimal representation of the raw byte. Exceptions from this rule are " and \, which are escaped by prepending a backslash, and all whitespace characters, which are written in their C-style notation (\n, \t, etc). In versions prior to 2.0.46, no escaping was performed on these strings so you had to be quite careful when dealing with raw log files.

Since httpd 2.0, unlike 1.3, the %b and %B format strings do not represent the number of bytes sent to the client, but simply the size in bytes of the HTTP response (which will differ, for instance, if the connection is aborted, or if SSL is used). The %O format provided by `mod_logio` will log the actual number of bytes sent over the network.

> Note: `mod_cache` is implemented as a quick-handler and not as a standard handler. Therefore, the %R format string will not return any handler information when content caching is involved.

Examples

Some commonly used log format strings are:

Common Log Format (CLF)
```
"%h %l %u %t \"%r\" %>s %b"
```

Common Log Format with Virtual Host
```
"%v %h %l %u %t \"%r\" %>s %b"
```

NCSA extended/combined log format
```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
```

Referer log format
```
"%{Referer}i -> %U"
```

Agent (Browser) log format
```
"%{User-agent}i"
```

You can use the %{format}t directive multiple times to build up a time format using the extended format tokens like msec_frac:

Timestamp including milliseconds
```
"%{%d/%b/%Y %T}t.%{msec_frac}t %{%z}t"
```

Security Considerations

See the security tips (↗ ../misc/security_tips.html#serverroot) document for details on why your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.

BufferedLogs Directive

Description:	Buffer log entries in memory before writing to disk
Syntax:	`BufferedLogs On|Off`
Default:	`BufferedLogs Off`
Context:	server config
Status:	Base
Module:	mod_log_config

The `BufferedLogs` directive causes `mod_log_config` to store several log entries in memory and write them together to disk, rather than writing them after each request. On some systems, this may result in more efficient disk access and hence higher performance. It may be set only once for the entire server; it cannot be configured per virtual-host.

> This directive should be used with caution as a crash might cause loss of logging data.

CustomLog Directive

Description:	Sets filename and format of log file

Syntax:	`CustomLog` *`file|pipe format|nickname`* `[env=[!]`*`environment-variable`*`| expr=`*`expression`*`]`
Context:	server config, virtual host
Status:	Base
Module:	mod_log_config

The `CustomLog` directive is used to log requests to the server. A log format is specified, and the logging can optionally be made conditional on request characteristics using environment variables.

The first argument, which specifies the location to which the logs will be written, can take one of the following two types of values:

file
: A filename, relative to the `ServerRoot`.

pipe
: The pipe character "`|`", followed by the path to a program to receive the log information on its standard input. See the notes on piped logs for more information.

> **Security:**
>
> If a program is used, then it will be run as the user who started `httpd`. This will be root if the server was started by root; be sure that the program is secure.

> **Note**
>
> When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashed are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.

The second argument specifies what will be written to the log file. It can specify either a *nickname* defined by a previous `LogFormat` directive, or it can be an explicit *format* string as described in the log formats (↗ #formats) section.

For example, the following two sets of directives have exactly the same effect:

```
# CustomLog with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "logs/access_log" common

# CustomLog with explicit format string
CustomLog "logs/access_log" "%h %l %u %t \"%r\" %>s %b"
```

The third argument is optional and controls whether or not to log a particular request. The condition can be the presence or absence (in the case of a '`env=!`*`name`*' clause) of a particular variable in the server environment (↗ ../env.html) . Alternatively, the condition can be expressed as arbitrary boolean expression (↗ ../expr.html) . If the condition is not satisfied, the request will not be logged. References to HTTP headers in the expression will not cause the header names to be added to the Vary header.

Environment variables can be set on a per-request basis using the `mod_setenvif` and/or `mod_rewrite` modules. For example, if you want to record requests for all GIF images on your server in a separate logfile but not in your main log, you can use:

```
SetEnvIf Request_URI \.gif$ gif-image
CustomLog "gif-requests.log" common env=gif-image
CustomLog "nongif-requests.log" common env=!gif-image
```

Or, to reproduce the behavior of the old RefererIgnore directive, you might use the following:

```
SetEnvIf Referer example\.com localreferer
CustomLog "referer.log" referer env=!localreferer
```

GlobalLog Directive

Description:	Sets filename and format of log file
Syntax:	`GlobalLog`*`file|pipe format|nickname`* `[env=[!]`*`environment-variable`*`| expr=`*`expression`*`]`
Context:	server config
Status:	Base
Module:	mod_log_config
Compatibility:	Available in Apache HTTP Server 2.4.19 and later

The `GlobalLog` directive defines a log shared by the main server configuration and all defined virtual hosts.

The `GlobalLog` directive is identical to the `CustomLog` directive, apart from the following differences:

- `GlobalLog` is not valid in virtual host context.
- `GlobalLog` is used by virtual hosts that define their own `CustomLog`, unlike a globally specified `CustomLog`.

LogFormat Directive

Description:	Describes a format for use in a log file
Syntax:	`LogFormat` *`format|nickname`* `[`*`nickname`*`]`
Default:	`LogFormat "%h %l %u %t \"%r\" %>s %b"`
Context:	server config, virtual host
Status:	Base
Module:	mod_log_config

This directive specifies the format of the access log file.

The `LogFormat` directive can take one of two forms. In the first form, where only one argument is specified, this directive sets the log format which will be used by logs specified in subsequent `TransferLog` directives. The single argument can specify an explicit *format* as discussed in the custom log formats (↗ #formats) section above. Alternatively, it can use a *nickname* to refer to a log format defined in a previous `LogFormat` directive as described below.

The second form of the `LogFormat` directive associates an explicit *format* with a *nickname*. This *nickname* can then be used in subsequent `LogFormat` or `CustomLog` directives rather than repeating the entire format string. A `LogFormat` directive that defines a nickname **does nothing else** -- that is, it *only* defines the nickname, it doesn't actually apply the

format and make it the default. Therefore, it will not affect subsequent `TransferLog` directives. In addition, `LogFormat` cannot use one nickname to define another nickname. Note that the nickname should not contain percent signs (%).

Example

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost_common
```

TransferLog Directive

Description:	Specify location of a log file
Syntax:	`TransferLog` *`file|pipe`*
Context:	server config, virtual host
Status:	Base
Module:	mod_log_config

This directive has exactly the same arguments and effect as the `CustomLog` directive, with the exception that it does not allow the log format to be specified explicitly or for conditional logging of requests. Instead, the log format is determined by the most recently specified `LogFormat` directive which does not define a nickname. Common Log Format is used if no other format has been specified.

Example

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
TransferLog logs/access_log
```

Comments

Notice:
This is not a Q&A section. Comments placed here should be pointed towards suggestions on improving the documentation or server, and may be removed by our moderators if they are either implemented or considered invalid/off-topic. Questions on how to manage the Apache HTTP Server should be directed at either our IRC channel, #httpd, on Libera.chat, or sent to our mailing lists.