# Environment Variables in Apache

There are two kinds of environment variables that affect the Apache HTTP Server.

First, there are the environment variables controlled by the underlying operating system. These are set before the server starts. They can be used in expansions in configuration files, and can optionally be passed to CGI scripts and SSI using the PassEnv directive.

Second, the Apache HTTP Server provides a mechanism for storing information in named variables that are also called *environment variables*. This information can be used to control various operations such as logging or access control. The variables are also used as a mechanism to communicate with external programs such as CGI scripts. This document discusses different ways to manipulate and use these variables.

Although these variables are referred to as *environment variables*, they are not the same as the environment variables controlled by the underlying operating system. Instead, these variables are stored and manipulated in an internal Apache structure. They only become actual operating system environment variables when they are provided to CGI scripts and Server Side Include scripts. If you wish to manipulate the operating system environment under which the server itself runs, you must use the standard environment manipulation mechanisms provided by your operating system shell.

## Setting Environment Variables

| Related Modules | Related Directives |
|---|---|
| mod_cache | BrowserMatch |
| mod_env | BrowserMatchNoCase |
| mod_rewrite | PassEnv |
| mod_setenvif | RewriteRule |
| mod_unique_id | SetEnv |
|  | SetEnvIf |
|  | SetEnvIfNoCase |
|  | UnsetEnv |

### Basic Environment Manipulation

The most basic way to set an environment variable in Apache is using the unconditional `SetEnv` directive. Variables may also be passed from the environment of the shell which started the server using the `PassEnv` directive.

### Conditional Per-Request Settings

For additional flexibility, the directives provided by `mod_setenvif` allow environment variables to be set on a per-request basis, conditional on characteristics of particular requests. For example, a variable could be set only when a specific browser (User-Agent) is making a request, or only when a specific Referer [sic] header is found. Even more flexibility is available through the `mod_rewrite`'s `RewriteRule` which uses the `[E=...]` option to set environment variables.

### Unique Identifiers

Finally, `mod_unique_id` sets the environment variable `UNIQUE_ID` for each request to a value which is guaranteed to be unique across "all" requests under very specific conditions.

### Standard CGI Variables

In addition to all environment variables set within the Apache configuration and passed from the shell, CGI scripts and SSI pages are provided with a set of environment variables containing meta-information about the request as required by the CGI specification.

### Some Caveats

- It is not possible to override or change the standard CGI variables using the environment manipulation directives.
- When `suexec` is used to launch CGI scripts, the environment will be cleaned down to a set of *safe* variables before CGI scripts are launched. The list of *safe* variables is defined at compile-time in `suexec.c`.
- For portability reasons, the names of environment variables may contain only letters, numbers, and the underscore character. In addition, the first character may not be a number. Characters which do not match this restriction will be replaced by an underscore when passed to CGI scripts and SSI pages.
- A special case are HTTP headers which are passed to CGI scripts and the like via environment variables (see below). They are converted to uppercase and only dashes are replaced with underscores; if the header contains any other (invalid) character, the whole header is silently dropped. See below for a workaround.
- The `SetEnv` directive runs late during request processing meaning that directives such as `SetEnvIf` and `RewriteCond` will not see the variables set with it.
- When the server looks up a path via an internal subrequest such as looking for a `DirectoryIndex` or generating a directory listing with `mod_autoindex`, per-request environment variables are *not* inherited in the subrequest. Additionally, `SetEnvIf` directives are not separately evaluated in the subrequest due to the API phases `mod_setenvif` takes action in.

## Using Environment Variables

| Related Modules | Related Directives |
|---|---|
| `mod_authz_host` | `Require` |
| `mod_cgi` | `CustomLog` |
| `mod_ext_filter` | `Deny` |
| `mod_headers` | `ExtFilterDefine` |
| `mod_include` | `Header` |
| `mod_log_config` | `LogFormat` |
| `mod_rewrite` | `RewriteCond` |
| | `RewriteRule` |

### CGI Scripts

One of the primary uses of environment variables is to communicate information to CGI scripts. As discussed above, the environment passed to CGI scripts includes standard meta-information about the request in addition to any variables set within the Apache configuration. For more details, see the CGI tutorial.

### SSI Pages

Server-parsed (SSI) documents processed by `mod_include`'s `INCLUDES` filter can print environment variables using the `echo` element, and can use environment variables in flow control elements to makes parts of a page conditional on characteristics of a request. Apache also provides SSI pages with the standard CGI environment variables as discussed above. For more details, see the SSI tutorial.

### Access Control

Access to the server can be controlled based on the value of environment variables using the `allow from env=` and `deny from env=` directives. In combination with `SetEnvIf`, this allows for flexible control of access to the server based on characteristics of the client. For example, you can use these directives to deny access to a particular browser (User-Agent).

### Conditional Logging

Environment variables can be logged in the access log using the `LogFormat` option `%e`. In addition, the decision on whether or not to log requests can be made based on the status of environment variables using the conditional form of the `CustomLog` directive. In combination with `SetEnvIf` this allows for flexible control of which requests are logged. For example, you can choose not to log requests for filenames ending in `gif`, or you can choose to only log requests from clients which are outside your subnet.

### Conditional Response Headers

The `Header` directive can use the presence or absence of an environment variable to determine whether or not a certain HTTP header will be placed in the response to the client. This allows, for example, a certain response header to be sent only if a corresponding header is received in the request from the client.

### External Filter Activation

External filters configured by `mod_ext_filter` using the `ExtFilterDefine` directive can by activated conditional on an environment variable using the `disableenv=` and `enableenv=` options.

### URL Rewriting

The `%{ENV:`*`variable`*`}` form of *TestString* in the `RewriteCond` allows `mod_rewrite`'s rewrite engine to make decisions conditional on environment variables. Note that the variables accessible in `mod_rewrite` without the `ENV:` prefix are not actually environment variables. Rather, they are variables special to `mod_rewrite` which cannot be accessed from other modules.

## Special Purpose Environment Variables

Interoperability problems have led to the introduction of mechanisms to modify the way Apache behaves when talking to particular clients. To make these mechanisms as flexible as possible, they are invoked by defining environment variables, typically with `BrowserMatch`, though `SetEnv` and `PassEnv` could also be used, for example.

### downgrade-1.0

This forces the request to be treated as a HTTP/1.0 request even if it was in a later dialect.

### force-gzip

If you have the `DEFLATE` filter activated, this environment variable will ignore the accept-encoding setting of your browser and will send compressed output unconditionally.

### force-no-vary

This causes any `Vary` fields to be removed from the response header before it is sent back to the client. Some clients don't interpret this field correctly; setting this variable can work around this problem. Setting this variable also implies **force-response-1.0**.

### force-response-1.0

This forces an HTTP/1.0 response to clients making an HTTP/1.0 request. It was originally implemented as a result of a problem with AOL's proxies. Some HTTP/1.0 clients may not behave correctly when given an HTTP/1.1 response, and this can be used to interoperate with them.

### gzip-only-text/html

When set to a value of "1", this variable disables the `DEFLATE` output filter provided by `mod_deflate` for content-types other than `text/html`. If you'd rather use statically compressed files, `mod_negotiation` evaluates the variable as well (not only for gzip, but for all encodings that differ from "identity").

### no-gzip

When set, the `DEFLATE` filter of `mod_deflate` will be turned off and `mod_negotiation` will refuse to deliver encoded resources.

### no-cache

*Available in versions 2.2.12 and later*

When set, `mod_cache` will not save an otherwise cacheable response. This environment variable does not influence whether a response already in the cache will be served for the current request.

### nokeepalive

This disables `KeepAlive` when set.

### prefer-language

This influences `mod_negotiation`'s behaviour. If it contains a language tag (such as `en`, `ja` or `x-klingon`), `mod_negotiation` tries to deliver a variant with that language. If there's no such variant, the normal negotiation process applies.

### redirect-carefully

This forces the server to be more careful when sending a redirect to the client. This is typically used when a client has a known problem handling redirects. This was originally implemented as a result of a problem with Microsoft's WebFolders software which has a problem handling redirects on directory resources via DAV methods.

### suppress-error-charset

*Available in versions after 2.0.54*

When Apache issues a redirect in response to a client request, the response includes some actual text to be displayed in case the client can't (or doesn't) automatically follow the redirection. Apache ordinarily labels this text according to the character set which it uses, which is ISO-8859-1.

However, if the redirection is to a page that uses a different character set, some broken browser versions will try to use the character set from the redirection text rather than the actual page. This can result in Greek, for instance, being incorrectly rendered.

Setting this environment variable causes Apache to omit the character set for the redirection text, and these broken browsers will then correctly use that of the destination page.

**Security note**

Sending error pages without a specified character set may allow a cross-site-scripting attack for existing browsers (MSIE) which do not follow the HTTP/1.1 specification and attempt to "guess" the character set from the content. Such browsers can be easily fooled into using the UTF-7 character set, and UTF-7 content from input data (such as the request-URI) will not be escaped by the usual escaping mechanisms designed to prevent cross-site-scripting attacks.

### force-proxy-request-1.0, proxy-nokeepalive, proxy-sendchunked, proxy-sendcl, proxy-chain-auth, proxy-interim-response, proxy-initial-not-pooled

These directives alter the protocol behavior of mod_proxy. See the mod_proxy and mod_proxy_http documentation for more details.

## Examples

### Passing broken headers to CGI scripts

Starting with version 2.4, Apache is more strict about how HTTP headers are converted to environment variables in mod_cgi and other modules: Previously any invalid characters in header names were simply translated to underscores. This allowed for some potential cross-site-scripting attacks via header injection (see Unusual Web Bugs, slide 19/20).

If you have to support a client which sends broken headers and which can't be fixed, a simple workaround involving mod_setenvif and mod_headers allows you to still accept these headers:

```
#
# The following works around a client sending a broken
# header.
#
SetEnvIfNoCase ^Accept.Encoding$ ^(.*)$ fix_accept_enco
RequestHeader set Accept-Encoding %{fix_accept_encoding
```

### Changing protocol behavior with misbehaving clients

Earlier versions recommended that the following lines be included in httpd.conf to deal with known client problems. Since the affected clients are no longer seen in the wild, this configuration is likely no-longer necessary.

```
#
# The following directives modify normal HTTP response
# The first directive disables keepalive for Netscape 2
# spoof it. There are known problems with these browser
# The second directive is for Microsoft Internet Explor
# which has a broken HTTP/1.1 implementation and does n
# support keepalive when it is used on 301 or 302 (redi
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 f

#
# The following directive disables HTTP/1.1 responses t
# are in violation of the HTTP/1.0 spec by not being ab
# basic 1.1 response.
#
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
```

### Do not log requests for images in the access log

This example keeps requests for images from appearing in the access log. It can be easily modified to prevent logging of particular directories, or to prevent logging of requests coming from particular hosts.

```
SetEnvIf Request_URI \.gif image-request
SetEnvIf Request_URI \.jpg image-request
SetEnvIf Request_URI \.png image-request
CustomLog "logs/access_log" common env=!image-request
```

## Prevent "Image Theft"

This example shows how to keep people not on your server from using images on your server as inline-images on their pages. This is not a recommended configuration, but it can work in limited circumstances. We assume that all your images are in a directory called `/web/images`.

```
SetEnvIf Referer "^http://www\.example\.com/" local_ref
# Allow browsers that do not send Referer info
SetEnvIf Referer "^$" local_referal
<Directory "/web/images">
    Require env local_referal
</Directory>
```

For more information about this technique, see the "Keeping Your Images from Adorning Other Sites" tutorial on ServerWatch.