Apache HTTP Server Version 2.4

Apache Module mod_alias

Description:	Provides for mapping different parts of the host filesystem in the document tree and for URL redirection
Status:	Base
Module Identifier:	alias_module
Source File:	mod_alias.c

Summary

The directives contained in this module allow for manipulation and control of URLs as requests arrive at the server. The `Alias` and `ScriptAlias` directives are used to map between URLs and filesystem paths. This allows for content which is not directly under the `DocumentRoot` served as part of the web document tree. The `ScriptAlias` directive has the additional effect of marking the target directory as containing only CGI scripts.

The `Redirect` directives are used to instruct clients to make a new request with a different URL. They are often used when a resource has moved to a new location.

When the `Alias`, `ScriptAlias` and `Redirect` directives are used within a `<Location>` or `<LocationMatch>` section, expression syntax (↗ ../expr.html) can be used to manipulate the destination path or URL.

`mod_alias` is designed to handle simple URL manipulation tasks. For more complicated tasks such as manipulating the query string, use the tools provided by `mod_rewrite`.

Order of Processing

Aliases and Redirects occurring in different contexts are processed like other directives according to standard merging rules (↗ ../sections.html#mergin) . But when multiple Aliases or Redirects occur in the same context (for example, in the same `<VirtualHost>` section) they are processed in a particular order.

First, all Redirects are processed before Aliases are processed, and therefore a request that matches a `Redirect` or `RedirectMatch` will never have Aliases applied. Second, the Aliases and Redirects are processed in the order they appear in the configuration files, with the first match taking precedence.

For this reason, when two or more of these directives apply to the same sub-path, you must list the most specific path first in order for all the directives to have an effect. For example, the following configuration will work as expected:

```
Alias "/foo/bar" "/baz"
Alias "/foo" "/gaq"
```

But if the above two directives were reversed in order, the `/foo` `Alias` would always match before the `/foo/bar` `Alias`, so the latter directive would be ignored.

When the `Alias`, `ScriptAlias` and `Redirect` directives are used within a `<Location>` or `<LocationMatch>` section, these directives will take precedence over any globally defined `Alias`, `ScriptAlias` and `Redirect` directives.

Alias Directive

Description:	Maps URLs to filesystem locations
Syntax:	`Alias [`*`URL-path`*`] `*`file-path`*`|`*`directory-path`*
Context:	server config, virtual host, directory
Status:	Base
Module:	mod_alias

The `Alias` directive allows documents to be stored in the local filesystem other than under the `DocumentRoot`. URLs with a (%-decoded) path beginning with *URL-path* will be mapped to local files beginning with *directory-path*. The *URL-path* is case-sensitive, even on case-insensitive file systems.

```
Alias "/image" "/ftp/pub/image"
```

A request for `http://example.com/image/foo.gif` would cause the server to return the file `/ftp/pub/image/foo.gif`. Only complete path segments are matched, so the above alias would not match a request for `http://example.com/imagefoo.gif`. For more complex matching using regular expressions, see the `AliasMatch` directive.

Note that if you include a trailing / on the *URL-path* then the server will require a trailing / in order to expand the alias. That is, if you use

```
Alias "/icons/" "/usr/local/apache/icons/"
```

then the URL `/icons` will not be aliased, as it lacks that trailing /. Likewise, if you omit the slash on the *URL-path* then you must also omit it from the *file-path*.

Note that you may need to specify additional `<Directory>` sections which cover the *destination* of aliases. Aliasing occurs before `<Directory>` sections are checked, so only the destination of aliases are affected. (Note however `<Location>` sections are run through once before aliases are performed, so they will apply.)

In particular, if you are creating an `Alias` to a directory outside of your `DocumentRoot`, you may need to explicitly permit access to the target directory.

```
Alias "/image" "/ftp/pub/image"
<Directory "/ftp/pub/image">
    Require all granted
</Directory>
```

Any number slashes in the *URL-path* parameter matches any number of slashes in the requested URL-path.

If the `Alias` directive is used within a `<Location>` or `<LocationMatch>` section the URL-path is omitted, and the file-path is interpreted using expression syntax (↗ ../expr.html) .
This syntax is available in Apache 2.4.19 and later.

```
<Location "/image">
    Alias "/ftp/pub/image"
</Location>
<LocationMatch "/error/(?<NUMBER>[0-9]+)">
```

```
    Alias "/usr/local/apache/errors/%{env:MATCH_NUMBER}.html"
</LocationMatch>
```

AliasMatch Directive

Description:	Maps URLs to filesystem locations using regular expressions
Syntax:	`AliasMatch` *`regex file-path|directory-path`*
Context:	server config, virtual host
Status:	Base
Module:	mod_alias

This directive is equivalent to `Alias`, but makes use of regular expressions (↗ ../glossary.html#regex) , instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to activate the `/icons` directory, one might use:

```
AliasMatch "^/icons(/|$)(.*)" "/usr/local/apache/icons$1$2"
```

The full range of regular expression (↗ ../glossary.html#regex) power is available. For example, it is possible to construct an alias with case-insensitive matching of the URL-path:

```
AliasMatch "(?i)^/image(.*)" "/ftp/pub/image$1"
```

One subtle difference between `Alias` and `AliasMatch` is that `Alias` will automatically copy any additional part of the URI, past the part that matched, onto the end of the file path on the right side, while `AliasMatch` will not. This means that in almost all cases, you will want the regular expression to match the entire request URI from beginning to end, and to use substitution on the right side.

In other words, just changing `Alias` to `AliasMatch` will not have the same effect. At a minimum, you need to add `^` to the beginning of the regular expression and add `(.*)$` to the end, and add `$1` to the end of the replacement.

For example, suppose you want to replace this with AliasMatch:

```
Alias "/image/" "/ftp/pub/image/"
```

This is NOT equivalent - don't do this! This will send all requests that have /image/ anywhere in them to /ftp/pub/image/:

```
AliasMatch "/image/" "/ftp/pub/image/"
```

This is what you need to get the same effect:

```
AliasMatch "^/image/(.*)$" "/ftp/pub/image/$1"
```

Of course, there's no point in using `AliasMatch` where `Alias` would work. `AliasMatch` lets you do more complicated things. For example, you could serve different kinds of files from different directories:

```
AliasMatch "^/image/(.*)\.jpg$" "/files/jpg.images/$1.jpg"
AliasMatch "^/image/(.*)\.gif$" "/files/gif.images/$1.gif"
```

Multiple leading slashes in the requested URL are discarded by the server before directives from this module compares against the requested URL-path.

Redirect Directive

Description:	Sends an external redirect asking the client to fetch a different URL
Syntax:	`Redirect [`*`status`*`] [`*`URL-path`*`] `*`URL`*
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_alias

The `Redirect` directive maps an old URL into a new one by asking the client to refetch the resource at the new location.

The old *URL-path* is a case-sensitive (%-decoded) path beginning with a slash. A relative path is not allowed.

The new *URL* may be either an absolute URL beginning with a scheme and hostname, or a URL-path beginning with a slash. In this latter case the scheme and hostname of the current server will be added.

Then any request beginning with *URL-path* will return a redirect request to the client at the location of the target *URL*. Additional path information beyond the matched *URL-path* will be appended to the target URL.

```
# Redirect to a URL on a different host
Redirect "/service" "http://foo2.example.com/service"

# Redirect to a URL on the same host
Redirect "/one" "/two"
```

If the client requests `http://example.com/service/foo.txt`, it will be told to access `http://foo2.example.com/service/foo.txt` instead. This includes requests with `GET` parameters, such as `http://example.com/service/foo.pl?q=23&a=42`, it will be redirected to `http://foo2.example.com/service/foo.pl?q=23&a=42`. Note that `POST`s will be discarded.
Only complete path segments are matched, so the above example would not match a request for `http://example.com/servicefoo.txt`. For more complex matching using the expression syntax (↗ ../expr.html) , omit the URL-path argument as described below. Alternatively, for matching using regular expressions, see the `RedirectMatch` directive.

> **Note**
>
> `Redirect` directives take precedence over `Alias` and `ScriptAlias` directives, irrespective of their ordering in the configuration file. `Redirect` directives inside a Location take precedence over `Redirect` and `Alias` directives with an *URL-path*.

If no *status* argument is given, the redirect will be "temporary" (HTTP status 302). This indicates to the client that the resource has moved temporarily. The *status* argument can be used to return other HTTP status codes:

permanent
: Returns a permanent redirect status (301) indicating that the resource has moved permanently.

temp
: Returns a temporary redirect status (302). This is the default.

seeother
: Returns a "See Other" status (303) indicating that the resource has been replaced.

gone
: Returns a "Gone" status (410) indicating that the resource has been permanently removed. When this status is used the *URL* argument should be omitted.

Other status codes can be returned by giving the numeric status code as the value of *status*. If the status is between 300 and 399, the *URL* argument must be present. If the status is *not* between 300 and 399, the *URL* argument must be omitted. The status must be a valid HTTP status code, known to the Apache HTTP Server (see the function `send_error_response` in http_protocol.c).

```
Redirect permanent "/one" "http://example.com/two"
Redirect 303 "/three" "http://example.com/other"
```

If the `Redirect` directive is used within a `<Location>` or `<LocationMatch>` section with the *URL-path* omitted, then the *URL* parameter will be interpreted using expression syntax (↗ ../expr.html) .
This syntax is available in Apache 2.4.19 and later.

```
<Location "/one">
    Redirect permanent "http://example.com/two"
</Location>
<Location "/three">
    Redirect 303 "http://example.com/other"
</Location>
<LocationMatch "/error/(?<NUMBER>[0-9]+)">
    Redirect permanent "http://example.com/errors/%{env:MATCH_NUMBER}.html"
</LocationMatch>
```

RedirectMatch Directive

Description:	Sends an external redirect based on a regular expression match of the current URL
Syntax:	`RedirectMatch [`*`status`*`] `*`regex URL`*
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_alias

This directive is equivalent to `Redirect`, but makes use of regular expressions (↗ ../glossary.html#regex) , instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to redirect all GIF files to like-named JPEG files on another server, one might use:

```
RedirectMatch "(.*)\.gif$" "http://other.example.com$1.jpg"
```

The considerations related to the difference between `Alias` and `AliasMatch` also apply to the difference between `Redirect` and `RedirectMatch`. See `AliasMatch` for details.

RedirectPermanent Directive

Description:	Sends an external permanent redirect asking the client to fetch a different URL
Syntax:	`RedirectPermanent `*`URL-path URL`*
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_alias

This directive makes the client know that the Redirect is permanent (status 301). Exactly equivalent to `Redirect permanent`.

RedirectTemp Directive

Description:	Sends an external temporary redirect asking the client to fetch a different URL
Syntax:	`RedirectTemp `*`URL-path URL`*
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_alias

This directive makes the client know that the Redirect is only temporary (status 302). Exactly equivalent to `Redirect temp`.

ScriptAlias Directive

Description:	Maps a URL to a filesystem location and designates the target as a CGI script
Syntax:	`ScriptAlias [`*`URL-path`*`] `*`file-path`*`|`*`directory-path`*
Context:	server config, virtual host, directory
Status:	Base
Module:	mod_alias

The `ScriptAlias` directive has the same behavior as the `Alias` directive, except that in addition it marks the target directory as containing CGI scripts that will be processed by `mod_cgi`'s cgi-script handler. URLs with a case-sensitive (%-decoded) path beginning with *URL-path* will be mapped to scripts beginning with the second argument, which is a full

pathname in the local filesystem.

```
ScriptAlias "/cgi-bin/" "/web/cgi-bin/"
```

A request for `http://example.com/cgi-bin/foo` would cause the server to run the script `/web/cgi-bin/foo`. This configuration is essentially equivalent to:

```
Alias "/cgi-bin/" "/web/cgi-bin/"
<Location "/cgi-bin">
    SetHandler cgi-script
    Options +ExecCGI
</Location>
```

`ScriptAlias` can also be used in conjunction with a script or handler you have. For example:

```
ScriptAlias "/cgi-bin/" "/web/cgi-handler.pl"
```

In this scenario all files requested in `/cgi-bin/` will be handled by the file you have configured, this allows you to use your own custom handler. You may want to use this as a wrapper for CGI so that you can add content, or some other bespoke action.

It is safer to avoid placing CGI scripts under the `DocumentRoot` in order to avoid accidentally revealing their source code if the configuration is ever changed. The `ScriptAlias` makes this easy by mapping a URL and designating CGI scripts at the same time. If you do choose to place your CGI scripts in a directory already accessible from the web, do not use `ScriptAlias`. Instead, use `<Directory>`, `SetHandler`, and `Options` as in:

```
<Directory "/usr/local/apache2/htdocs/cgi-bin">
    SetHandler cgi-script
    Options ExecCGI
</Directory>
```

This is necessary since multiple *URL-paths* can map to the same filesystem location, potentially bypassing the `ScriptAlias` and revealing the source code of the CGI scripts if they are not restricted by a `Directory` section.

If the `ScriptAlias` directive is used within a `<Location>` or `<LocationMatch>` section with the URL-path omitted, then the URL parameter will be interpreted using expression syntax (↗ ../expr.html) .
This syntax is available in Apache 2.4.19 and later.

```
<Location "/cgi-bin">
    ScriptAlias "/web/cgi-bin/"
</Location>
<LocationMatch "/cgi-bin/errors/(?<NUMBER>[0-9]+)">
    ScriptAlias "/web/cgi-bin/errors/%{env:MATCH_NUMBER}.cgi"
</LocationMatch>
```

See also

- CGI Tutorial

ScriptAliasMatch Directive

Description:	Maps a URL to a filesystem location using a regular expression and designates the target as a CGI script
Syntax:	`ScriptAliasMatch` *`regex file-path|directory-path`*
Context:	server config, virtual host
Status:	Base
Module:	mod_alias

This directive is equivalent to `ScriptAlias`, but makes use of regular expressions (↗ ../glossary.html#regex) , instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to activate the standard `/cgi-bin`, one might use:

```
ScriptAliasMatch "^/cgi-bin(.*)" "/usr/local/apache/cgi-bin$1"
```

As for AliasMatch, the full range of regular expression (↗ ../glossary.html#rexex) power is available. For example, it is possible to construct an alias with case-insensitive matching of the URL-path:

```
ScriptAliasMatch "(?i)^/cgi-bin(.*)" "/usr/local/apache/cgi-bin$1"
```

The considerations related to the difference between `Alias` and `AliasMatch` also apply to the difference between `ScriptAlias` and `ScriptAliasMatch`. See `AliasMatch` for details.

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.