



| | [FAQ](#) | |

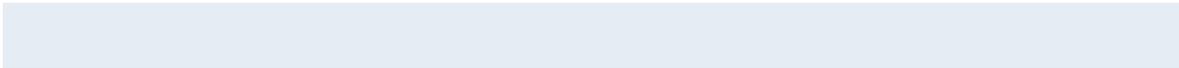


HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#)

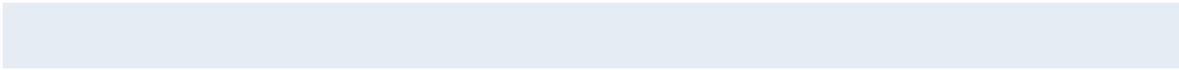
# Apache HTTP Server Version 2.4





2.0

1.3 2.0



(MPM)



(content negotiation)

(DSO)

URL

SSL/TLS

CGI Suexec

URL (rewriting)

## How-To /

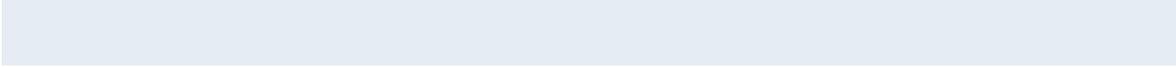
[...](#)

[CGI:](#)

[.htaccess](#)

[Server Side Includes \(SSI\)](#)

[\(public\\_html\)](#)



[Microsoft Windows](#)

[Novell NetWare](#)

[EBCDIC](#)

(FAQ)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## Upgrading to 2.4 from 2.2

In order to assist folks upgrading, we maintain a document describing information critical to existing Apache HTTP Server users. These are intended to be brief notes, and you should be able to find more information in either the [New Features](#) document, or in the `src/CHANGES` file. Application and module developers can find a summary of API changes in the [API updates](#) overview.

This document describes changes in server behavior that might require you to change your configuration or how you use the server in order to continue using 2.4 as you are currently using 2.2. To take advantage of new features in 2.4, see the New Features document.

This document describes only the changes from 2.2 to 2.4. If you are upgrading from version 2.0, you should also consult the [2.0 to 2.2 upgrading document](#).



### See also

[Overview of new features in Apache HTTP Server 2.4](#)



## Compile-time Configuration Changes

The compilation process is very similar to the one used in version 2.2. Your old `configure` command line (as found in `build/config.nice` in the installed server directory) can be used in most cases. There are some changes in the default settings. Some details of changes:

- These modules have been removed: `mod_authn_default`, `mod_authz_default`, `mod_mem_cache`. If you were using `mod_mem_cache` in 2.2, look at [mod\\_cache\\_disk](#) in 2.4.
- All load balancing implementations have been moved to individual, self-contained `mod_proxy` submodules, e.g. [mod\\_lbmethod\\_bybusyness](#). You might need to build and load any of these that your configuration uses.
- Platform support has been removed for BeOS, TPF, and even older platforms such as A/UX, Next, and Tandem. These were believed to be broken anyway.
- `configure`: dynamic modules (DSO) are built by default
- `configure`: By default, only a basic set of modules is loaded. The other `LoadModule` directives are commented out in the configuration file.
- `configure`: the "most" module set gets built by default
- `configure`: the "reallyall" module set adds developer modules to the "all" set



There have been significant changes in authorization configuration, and other minor configuration changes, that could require changes to your 2.2 configuration files before using them for 2.4.

### Authorization

Any configuration file that uses authorization will likely need changes.

You should review the [Authentication, Authorization and Access Control Howto](#), especially the section [Beyond just authorization](#) which explains the new mechanisms for controlling the order in which the authorization directives are applied.

Directives that control how authorization modules respond when they don't match the authenticated user have been removed: This includes `AuthzLDAPAuthoritative`, `AuthzDBDAuthoritative`, `AuthzDBMAuthoritative`, `AuthzGroupFileAuthoritative`, `AuthzUserAuthoritative`, and `AuthzOwnerAuthoritative`. These directives have been replaced by the more expressive [RequireAny](#), [RequireNone](#), and [RequireAll](#).

If you use `mod_authz_dbm`, you must port your configuration to use `Require dbm-group . . .` in place of `Require group . . .`

### Access control

In 2.2, access control based on client hostname, IP address, and other characteristics of client requests was done using the directives [Order](#), [Allow](#), [Deny](#), and [Satisfy](#).

In 2.4, such access control is done in the same way as other authorization checks, using the new module `mod_authz_host`.

The old access control idioms should be replaced by the new authentication mechanisms, although for compatibility with old configurations, the new module [mod\\_access\\_compat](#) is provided.

### Mixing old and new directives

Mixing old directives like [Order](#), [Allow](#) or [Deny](#) with new ones like [Require](#) is technically possible but discouraged.

[mod\\_access\\_compat](#) was created to support configurations containing only old directives to facilitate the 2.4 upgrade. Please check the examples below to get a better idea about issues that might arise.

Here are some examples of old and new ways to do the same access control.

In this example, there is no authentication and all requests are denied.

#### 2.2 configuration:

```
Order deny,allow  
Deny from all
```

#### 2.4 configuration:

```
Require all denied
```

In this example, there is no authentication and all requests are allowed.

#### 2.2 configuration:

```
Order allow,deny  
Allow from all
```

## 2.4 configuration:

```
Require all granted
```

In the following example, there is no authentication and all hosts in the example.org domain are allowed access; all other hosts are denied access.

## 2.2 configuration:

```
Order Deny,Allow
Deny from all
Allow from example.org
```

## 2.4 configuration:

```
Require host example.org
```

In the following example, mixing old and new directives leads to unexpected results.

## Mixing old and new directives: NOT WORKING AS EXPECTED

```
DocumentRoot "/var/www/html"
```

```
<Directory "/">
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
```

```
<Location "/server-status">
    SetHandler server-status
    Require local
</Location>
```

```
access.log - GET /server-status 403 127.0.0.1
```

```
error.log - AH01797: client denied by server configuration: /var
```

Why httpd denies access to servers-status even if the

configuration seems to allow it? Because [mod\\_access\\_compat](#) directives take precedence over the [mod\\_authz\\_host](#) one in this configuration [merge](#) scenario.

This example conversely works as expected:

### Mixing old and new directives: WORKING AS EXPECTED

```
DocumentRoot "/var/www/html"

<Directory "/">
    AllowOverride None
    Require all denied
</Directory>

<Location "/server-status">
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow From 127.0.0.1
</Location>

access.log - GET /server-status 200 127.0.0.1
```

So even if mixing configuration is still possible, please try to avoid it when upgrading: either keep old directives and then migrate to the new ones on a later stage or just migrate everything in bulk.

In many configurations with authentication, where the value of the [Satisfy](#) was the default of *ALL*, snippets that simply disabled host-based access control are omitted:

### 2.2 configuration:

```
Order Deny,Allow
Deny from all
AuthBasicProvider File
AuthUserFile /example.com/conf/users.passwd
AuthName secure
Require valid-user
```

## 2.4 configuration:

```
# No replacement needed
AuthBasicProvider File
AuthUserFile /example.com/conf/users.passwd
AuthName secure
Require valid-user
```

In configurations where both authentication and access control were meaningfully combined, the access control directives should be migrated. This example allows requests meeting *both* criteria:

## 2.2 configuration:

```
Order allow,deny
Deny from all
# Satisfy ALL is the default
Satisfy ALL
Allow from 127.0.0.1
AuthBasicProvider File
AuthUserFile /example.com/conf/users.passwd
AuthName secure
Require valid-user
```

## 2.4 configuration:

```
AuthBasicProvider File
AuthUserFile /example.com/conf/users.passwd
AuthName secure
<RequireAll>
  Require valid-user
  Require ip 127.0.0.1
</RequireAll>
```

In configurations where both authentication and access control were meaningfully combined, the access control directives should be migrated. This example allows requests meeting *either* criteria:

## 2.2 configuration:

```
Order allow,deny
Deny from all
Satisfy any
```

```
Allow from 127.0.0.1
AuthBasicProvider File
AuthUserFile /example.com/conf/users.passwd
AuthName secure
Require valid-user
```

## 2.4 configuration:

```
AuthBasicProvider File
AuthUserFile /example.com/conf/users.passwd
AuthName secure
# Implicitly <RequireAny>
Require valid-user
Require ip 127.0.0.1
```

## Other configuration changes

Some other small adjustments may be necessary for particular configurations as discussed below.

- `MaxRequestsPerChild` has been renamed to `MaxConnectionsPerChild`, describes more accurately what it does. The old name is still supported.
- `MaxClients` has been renamed to `MaxRequestWorkers`, which describes more accurately what it does. For async MPMs, like `event`, the maximum number of clients is not equivalent than the number of worker threads. The old name is still supported.
- The `DefaultType` directive no longer has any effect, other than to emit a warning if it's used with any value other than none. You need to use other configuration settings to replace it in 2.4.
- `AllowOverride` now defaults to None.
- `EnableSendfile` now defaults to Off.
- `FileETag` now defaults to "MTime Size" (without INode).
- `mod_dav_fs`: The format of the `DavLockDB` file has changed for systems with inodes. The old `DavLockDB` file must be

deleted on upgrade.

- [KeepAlive](#) only accepts values of On or Off. Previously, any value other than "Off" or "0" was treated as "On".
- Directives AcceptMutex, LockFile, RewriteLock, SSLMutex, SSLStaplingMutex, and WatchdogMutexPath have been replaced with a single [Mutex](#) directive. You will need to evaluate any use of these removed directives in your 2.2 configuration to determine if they can just be deleted or will need to be replaced using [Mutex](#).
- [mod\\_cache](#): [CacheIgnoreURLSessionIdentifiers](#) now does an exact match against the query string instead of a partial match. If your configuration was using partial strings, e.g. using sessionid to match `/someapplication/image.gif;jsessionid=12345678` then you will need to change to the full string `jsessionid`.
- [mod\\_cache](#): The second parameter to [CacheEnable](#) only matches forward proxy content if it begins with the correct protocol. In 2.2 and earlier, a parameter of '/' matched all content.
- [mod\\_ldap](#): [LDAPTrustedClientCert](#) is now consistently a per-directory setting only. If you use this directive, review your configuration to make sure it is present in all the necessary directory contexts.
- [mod\\_filter](#): [FilterProvider](#) syntax has changed and now uses a boolean expression to determine if a filter is applied.
- [mod\\_include](#):
  - The `#if expr` element now uses the new [expression parser](#). The old syntax can be restored with the new directive [SSILegacyExprParser](#).
  - An SSI\* config directive in directory scope no longer causes all other per-directory SSI\* directives to be reset to their default values.

- [mod\\_charset\\_lite](#): The `LogLevel` option has been removed in favour of per-module `LogLevel` configuration.
- [mod\\_ext\\_filter](#): The `LogLevel` option has been removed in favour of per-module `LogLevel` configuration.
- [mod\\_proxy\\_scgi](#): The default setting for `PATH_INFO` has changed from `httpd 2.2`, and some web applications will no longer operate properly with the new `PATH_INFO` setting. The previous setting can be restored by configuring the `proxy-scgi-pathinfo` variable.
- [mod\\_ssl](#): CRL based revocation checking now needs to be explicitly configured through `SSLCARevocationCheck`.
- [mod\\_substitute](#): The maximum line length is now limited to 1MB.
- [mod\\_reqtimeout](#): If the module is loaded, it will now set some default timeouts.
- [mod\\_dumpio](#): `DumpIOLogLevel` is no longer supported. Data is always logged at `LogLevel` `trace7`.
- On Unix platforms, piped logging commands configured using either `ErrorLog` or `CustomLog` were invoked using `/bin/sh -c` in 2.2 and earlier. In 2.4 and later, piped logging commands are executed directly. To restore the old behaviour, see the [piped logging documentation](#).



- [mod\\_autoindex](#): will now extract titles and display descriptions for .xhtml files, which were previously ignored.
- [mod\\_ssl](#): The default format of the \*\_DN variables has changed. The old format can still be used with the new LegacyDNStringFormat argument to [SSLOptions](#). The SSLv2 protocol is no longer supported. [SSLProxyCheckPeerCN](#) and [SSLProxyCheckPeerExpire](#) now default to On, causing proxy requests to HTTPS hosts with bad or outdated certificates to fail with a 502 status code (Bad gateway)
- [htpasswd](#) now uses MD5 hash by default on all platforms.
- The [NameVirtualHost](#) directive no longer has any effect, other than to emit a warning. Any address/port combination appearing in multiple virtual hosts is implicitly treated as a name-based virtual host.
- [mod\\_deflate](#) will now skip compression if it knows that the size overhead added by the compression is larger than the data to be compressed.
- Multi-language error documents from 2.2.x may not work unless they are adjusted to the new syntax of [mod\\_include](#)'s #if expr= element or the directive [SSILegacyExprParser](#) is enabled for the directory containing the error documents.
- The functionality provided by [mod\\_authn\\_alias](#) in previous versions (i.e., the [AuthnProviderAlias](#) directive) has been moved into [mod\\_authn\\_core](#).
- The RewriteLog and RewriteLogLevel directives have been removed. This functionality is now provided by configuring the appropriate level of logging for the [mod\\_rewrite](#) module using the [LogLevel](#) directive. See also the [mod\\_rewrite logging](#) section.



## Third-Party Modules

All modules must be recompiled for 2.4 before being loaded.

Many third-party modules designed for version 2.2 will otherwise work unchanged with the Apache HTTP Server version 2.4. Some will require changes; see the [API update](#) overview.



- ## Common problems when upgrading
- Startup errors:
    - Invalid command 'User', perhaps misspelled or defined by a module not included in the server configuration - load module [mod\\_unixd](#)
    - Invalid command 'Require', perhaps misspelled or defined by a module not included in the server configuration, or Invalid command 'Order', perhaps misspelled or defined by a module not included in the server configuration - load module [mod\\_access\\_compat](#), or update configuration to 2.4 authorization directives.
    - Ignoring deprecated use of DefaultType in line NN of /path/to/httpd.conf - remove [DefaultType](#) and replace with other configuration settings.
    - Invalid command 'AddOutputFilterByType', perhaps misspelled or defined by a module not included in the server configuration - [AddOutputFilterByType](#) has moved from the core to [mod\\_filter](#), which must be loaded.
  - Errors serving requests:
    - configuration error: couldn't check user: /path - load module [mod\\_authn\\_core](#).
    - .htaccess files aren't being processed - Check for an appropriate [AllowOverride](#) directive; the default changed to None in 2.4.

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# Apache 2.0

1.3 2.0



[1.3 2.0](#)



---

POSIX  
(scalability) .

autoconf libtool .

[mod\\_echo](#) .

Apache 2.0 BeOS, OS/2,  
POSIX API  
(MPM) Apache Portable Runtime (APR) .

### API

API 2.0 . 1.3 . 2.0  
, (hook) . ,

### IPv6

Apache Portable Runtime IPv6 IPv6  
, [Listen, NameVirtualHost, VirtualHost](#)  
. (, "Listen [2001:db8::1]:8080").

INCLUDES CGI Server Side Include

[mod\\_ext\\_filter](#) CGI

SSI .

. Port BindAdd  
Listen . ServerName

### Windows NT

Windows NT Apache 2.0 utf-8 .  
, Windows 2000 Windows XP Windows  
NT . Windows 95, 98, ME ,

### Updated

Apache 2.0 [Perl \(Perl Compatible Regular Expression Library\)](#) (PCRE) . Perl 5 .



---

## mod\_ssl

Apache 2.0 . OpenSSL SSL/TLS

## mod\_dav

Apache 2.0 . HTTP Distributed  
Authoring and Versioning (DAV) .

## mod\_deflate

Apache 2.0 .

## mod\_auth\_ldap

Apache 2.0.41 . HTTP Basic Authentic  
LDAP . [mod\\_ldap](#) (cc

## mod\_auth\_digest

## mod\_charset\_lite

Apache 2.0 .

## mod\_file\_cache

Apache 2.0 . Apache 1.3 mod\_mmap\_s

## mod\_headers

Apache 2.0 . [mod\\_proxy](#)

## mod\_proxy

HTTP/1.1 .

`<Proxy>` ( ) .

`<Directory "proxy:...">` .

proxy\_connect, proxy\_ftp, proxy\_http

## mod\_negotiation

[ForceLanguagePriority](#) NOT ACCEPTABLE  
MULTIPLE CHOICES .  
MultiViews ,  
map .

### **mod\_autoindex**

HTML ,

### **mod\_include**

SSI , SSI  
. mod\_include ( Perl )  
[mod\\_include](#) \$0 ... \$9 .

### **mod\_auth\_dbm**

[AuthDBMType](#) DBM .



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# The Apache License, Version 2.0

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

## TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

### 1. Definitions

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not

limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by

Licensor and subsequently incorporated within the Work.

2. **Grant of Copyright License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. **Grant of Patent License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. **Redistribution.** You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - a. You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - b. You must cause any modified files to carry prominent notices stating that You changed the files; and

- c. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- d. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall

supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity,

or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

## END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]
```

```
Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at
```

```
http://www.apache.org/licenses/LICENSE-2.0
```

```
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or impl  
See the License for the specific language governing permissions and  
limitations under the License.
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



2.0 1.3 .  
2.0

1.3 .  
libtool auto

(,

2.0.50 2.0.51), .



```
$ lynx http://httpd.apache.org/download.cgi
$ gzip -d httpd-2_1_2N.tar.gz
$ tar xvf httpd-2_1_2N.tar
$ ./configure --prefix=PREFIX
$ make
$ make install
$ vi PREFIX/conf/httpd.conf
$ PREFIX/bin/apachectl start
```

*NN* ,      *PREFIX* .      *PRE*  
          /usr/local/apache2 .



:

50 MB .

10 MB .

## ANSI-C

ANSI-C . [Free Software Foundation \(FSF\) GNU C compiler \(GCC\)](#) . ( 2.7.2 .) GCC

ANSI . PATH make .

HTTP .

Network Time Protocol (NTP)

ntpdate xnt

. NTP

[comp.prc](#)

[NTP](#) .

## Perl 5 []

(Perl) [apxs dbmmanage](#) Perl 5 .

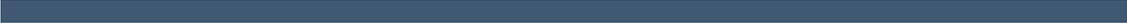
( 5.003 .) ` configure' 2.0

. Perl (

Perl 4 Perl 5) ./configure --

perl () .





. [\[redacted\]](#) .  
( ) , .  
INSTALL.bindist

. [\[redacted\]](#) , [PGP \[redacted\]](#) .



---

tar :

```
$ gzip -d httpd-2_1_MN.tar.gz  
$ tar xvf httpd-2_1_MN.tar
```



```

configure .(
libtool,
.)

CVS
buildconf

./configure

./configure

Base
module module mod_
--enable-module=shared
object, DSO) . , --disable-module Base
configure

configure ,
configure . configure manp

DSO
mod rewrite mod spelling /sw/pkg/apache
:

```

```

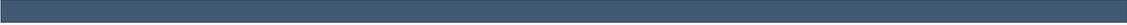
$ CC="pgcc" CFLAGS="-O2" \
./configure --prefix=/sw/pkg/apache \
--enable-rewrite=shared \
--enable-speling=shared

```

configure Makefile

configure [configure manpage](#) .





:

\$ make

. III/ 2.2

3 .

.



```
( --prefix )
```

```
PREFIX :
```

```
$ make install
```



*PREFIX/conf/* .

```
$ vi PREFIX/conf/httpd.conf
```

<http://httpd.apache.org/docs/2.4/> [docs/manual/](#) .



---

:

```
$ PREFIX/bin/apachectl start
```

```
URL http://localhost/ .  
PREFIX/htdocs/ DocumentRoot .
```

```
$ PREFIX/bin/apachectl stop
```



```

, 1.3 2.0 2.0 2.2 )
. API .
(, 2.0.55 2.0.57) . ma
, , ,
. configure ,
.( 2.0.41 .
, .
configure .
config.nice , , :

```

```

$ ./config.nice
$ make
$ make install
$ PREFIX/bin/apachectl stop
$ PREFIX/bin/apachectl start

```

```

prefix( Listen )
.

```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

[Empty rectangular box]



Windows NT, 2000, XP ,

Windows 95 ME .

[Grey rectangular box]

[Grey rectangular box]

[httpd](#)

h



[Grey rectangular box]

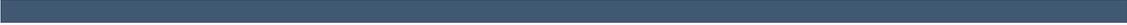
[httpd](#)

[apachectl](#)



```
Listen 80(          1024 )
,
httpd root ,
.
apachectl          httpd
apachectl .        apachectl ,
apachectl.,       apachectl          HTTP
.
httpd httpd.conf . ,
-f .
/usr/local/apache2/bin/apachectl -f
/usr/local/apache2/conf/httpd.conf
,
DocumentRoot      ()
```



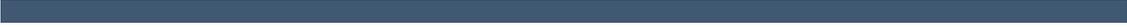


```
,  
    "    Unable to bind to Port ...".  
:
```

- root .
- .

[FAQ](#) .





```
, ( rc.  
apachectl . root .  
.  
apachectl SysV init  
restart, stop httpd . a  
init . .
```



[httpd](#) [apachectl](#),

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



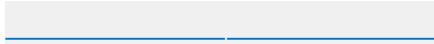
HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



. NT, 2000, XP

ME



[httpd](#)  
[apachectl](#)



```
kill -TERM httpd .
pid (signal) . ,
. , TERM, HUP, USR1.
:
```

```
kill -TERM `cat /usr/local/apache2/logs/httpd.pid`
```

```
httpd -k .
restart, graceful httpd .
apachectl .
```

```
httpd , :
```

```
tail -f /usr/local/apache2/logs/error_log
```

```
ServerRoot PidFile .
```



---

**: TERM**

apachectl -k stop

TERM stop

.

.

,

.



```
: USR1
    apachectl -k graceful
```

```
USR1 graceful
    )
```

```
(graceful restart) USR1 ( WINCH )
    apachectl graceful .
```

```
MPM
    StartServers, StartServers
StartServers . ,
StartServers
```

```
mod_status USR1 0 (
    )
    scoreboard .
```

```
status (
    USR1
    10
    15 .
```

```
( " ".)
    -t ( httpd )
    root
root ( httpd )
```



---

: HUP

apachectl -k restart

HUP restart

TERM

mod\_status HUP 0



Apache 1.2b9

(race condition) . (

, .) ""

ScoreBoardFile scoreboard

"bind: Address already in use" (USR1) "long lost child came home!" . , scoreboard

scoreboard . ScoreBoa

HTTP (KeepAlive)

. 1.2

KeepAlive

20



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

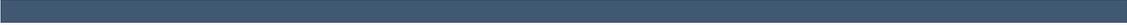
[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

--



.





```
mod mime <IfDefine>  
    Include  
    TypesConfig
```

```
httpd.conf .  
-f .  
Include  
mime .  
TypesConfig ,
```



---

```
. "\" .  
. .  
, "#"  
. , (indent)  
. .  
apachectl configtest -t  
. .
```



```
mod_so <IfModule>  
LoadModule
```

[base](#) .

-1 .

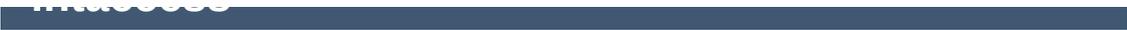


```
<Directory>
<DirectoryMatch>
<Files>
<FilesMatch>
<Location>
<LocationMatch>
<VirtualHost>
```

<DirectoryMatch>, <Files>, <FilesMatch>, <Location>,  
<LocationMatch> .

Directory, Location, Files .





```

AccessFileName
AllowOverride

```

```

() .
AccessFileName . .htaccess
. .htaccess
.
.htaccess
AllowOverride .htaccess
.htaccess .htaccess .

```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



, , , , URL .

.htaccess .



<u>core</u>	<u>&lt;Directory&gt;</u>
<u>mod_proxy</u>	<u>&lt;DirectoryMatch&gt;</u>
	<u>&lt;Files&gt;</u>
	<u>&lt;FilesMatch&gt;</u>
	<u>&lt;IfDefine&gt;</u>
	<u>&lt;IfModule&gt;</u>
	<u>&lt;Location&gt;</u>
	<u>&lt;LocationMatch&gt;</u>
	<u>&lt;Proxy&gt;</u>
	<u>&lt;ProxyMatch&gt;</u>
	<u>&lt;VirtualHost&gt;</u>

```

. . .
<IfDefine> <IfModule> .
. . .

```

```

<IfDefine> httpd .
,          httpd -DClosedForNow
:

```

```

<IfDefine ClosedForNow>
Redirect / http://otherserver.example.com/
</IfDefine>

```

```

<IfModule>

```

```

.
.
.

```

```

mod_mime_magic MimeMagicFiles .

```

```

<IfModule mod_mime_magic.c>

```

```
MimeMagicFile conf/magic
</IfModule>
```

```
<u><IfDefine> <IfModule> "!" . ,
```

.



```

(webospace) .
. . ,
        /usr/local/apache2,      "c:/Progra
Files/Apache Group/Apache2" .( ,
, .) .
        /dir/
/usr/local/apache2/htdocs/dir/ .
.

```

```

<Directory> <Files> .
<Directory> .          .htaccess
. , (index)           /var/web/d:
(index) .

```

```

<Directory /var/web/dir1>
Options +Indexes
</Directory>

```

```

<Files> .
,          private.html .

```

```

<Files private.html>
Order allow,deny
Deny from all
</Files>

```

```

        <Files> <Directory> .
,          /var/web/dir1/private.html,
/var/web/dir1/subdir2/private.html,
/var/web/dir1/subdir3/private.html
/var/web/dir1/          private.html .

```

```
<Directory /var/web/dir1>
<Files private.html>
Order allow,deny
Deny from all
</Files>
</Directory>
```

<Location> .  
 , /private URL- .  
http://yoursite.example.com/private,  
http://yoursite.example.com/private123,  
http://yoursite.example.com/private/dir/file.html  
 /private .

```
<Location /private>
Order Allow,Deny
Deny from all
</Location>
```

<Location> . URL  
mod\_status . server-status  
 .

```
<Location /server-status>
SetHandler server-status
</Location>
```

<Directory>, <Files>, <Location> C fnmatch  
 . "\*" , "?"  
 , "[ seq]" seq . "/" .  
 .  
 perl <DirectoryMatch>, <FilesMatch>,

<LocationMatch> .

.  
:

```
<Directory /home/*/public_html>  
Options Indexes  
</Directory>
```

:

```
<FilesMatch \.(?i:gif|jpe?g|png)$>  
Order allow,deny  
Deny from all  
</FilesMatch>
```

<Directory> <Files> . ( )

<Location> .

<Location> . (URL)

, . :

```
<Location /dir/>  
Order allow,deny  
Deny from all  
</Location>
```

http://yoursite.example.com/dir/ .

?

http://yoursit

.

<Directory>

.( . .

<Directory> .

Options

.)

.  
.  
.

<Location /> URL



---

<VirtualHost> .



<Proxy> <ProxyMatch> URL

mod\_proxy

cnn.cc

```
<Proxy http://cnn.com/*>  
Order allow,deny  
Deny from all  
</Proxy>
```



---

. <Directory>  
<DirectoryMatch>, <Files>, <FilesMatch>, <Location>,  
<LocationMatch>, <Proxy>, <ProxyMatch> . ,

:

- AllowOverride <Directory> .
- FollowSymLinks, SymLinksIfOwnerMatch, Options  
<Directory> .htaccess .
- Options <Files> <FilesMatch> .



```

.
.
:
1. ()          <Directory> .htaccess (
    .htaccess <Directory> )
2. <DirectoryMatch>( <Directory ~>)
3. <Files> <FilesMatch>
4. <Location> <LocationMatch>

<Directory>          . ( 1)
<Directory>          . ,
<Directory /var/web/dir> <Directory
/var/web/dir/subdir> .          <D
    <Include          <Incl
.
.

<VirtualHost>          .
.

mod_proxy , <Proxy>          <Directory> .
.

```

```

    <Location>/<LocationMatch> (Aliases
DocumentRoot          URL ) .
.

```

. A > B > C > D > E

```
<Location />
E
</Location>

<Files f.html>
D
</Files>

<VirtualHost *>
<Directory /a/b>
B
</Directory>
</VirtualHost>

<DirectoryMatch "^.*b$">
C
</DirectoryMatch>

<Directory /a/b>
A
</Directory>
```

. <Location> <Directory>

., !

```
<Location />
Order deny,allow
Allow from all
</Location>

# ! <Directory>
<Directory />
Order allow,deny
Allow from all
Deny from badguy.example.com
</Directory>
```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



core



---

<u>ServerName</u>	
<u>ServerAdmin</u>	
<u>ServerSignature</u>	
<u>ServerTokens</u>	
<u>UseCanonicalName</u>	

ServerAdmin ServerTokens

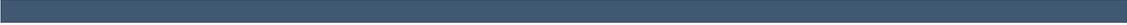
. ServerTokens HTTP .

ServerName UseCanonicalName URL .

,

.





- CoreDumpDirectory
- DocumentRoot
- ErrorLog
- LockFile
- PidFile
- ScoreBoardFile
- ServerRoot

. root

. (/)

.



```
LimitRequestBody
LimitRequestFields
LimitRequestFieldsize
LimitRequestLine
RLimitCPU
RLimitMEM
RLimitNPROC
ThreadStackSize
```

LimitRequest\* .  
of service) .

RLimit\* . CGI  
.

ThreadStackSize Netware .



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

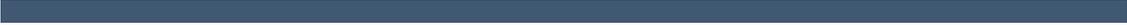
[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



.

.





( root) uid

.      .

, .



(error\_log)

```

ErrorLog
LogLevel

```

ErrorLog

```

(
    syslog error_log, OS/2

```

```

[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client
denied by server configuration:
/export/home/live/ap/htdocs/test

```

```

IP
( )
.CGI stderr .CGI
, 403
:

```

```

tail -f error_log

```



(Access Log)

<u>mod_log_config</u>	<u>CustomLog</u>
<u>mod_setenvif</u>	<u>LogFormat</u>
	<u>SetEnvIf</u>

```

CustomLog
LogFormat
Open Directory
mod_log_referer, mod_log_agent, CustomLog
CustomLog
C printf(1)
mod_log_config

```

### Common

```

LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common

```

```

common
(
" \n", " \t"

```

CustomLog  
ServerRoot

(Common Log Format, CLF)

CLF :

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

**127.0.0.1 (%h)**

( ) IP

[HostnameLook](#)

IP

[logresolve](#)

**- (%l)**

""

RFC 1413

[IdentityCheck](#) On

**frank (%u)**

HTTP

userid. CGI

REMOTE\_USER .

401 ( )

**[10/Oct/2000:13:55:36 -0700] (%t)**

[day/month/year:hour:minute:second zone]

day = 2

month = 3

year = 4

hour = 2

minute = 2

second = 2

zone = ('+' | '-' ) 4



```
"http://www.example.com/start.html" (\ "%  
{Referer}i\")
```

```
    "Referer" () HTTP .  
    /apache_pb.gif .)
```

```
"Mozilla/4.08 [en] (Win98; I ;Nav)" (\ "%  
{User-agent}i\")
```

```
    User-Agent HTTP .
```

```
    CustomLog . ,  
    . CLF , referer  
    ReferLog AgentLog .
```

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common  
CustomLog logs/access_log common  
CustomLog logs/referer_log "%{Referer}i -> %U"  
CustomLog logs/agent_log "%{User-agent}i"
```

```
, LogFormat . Cu
```

```
env= . : SetEnvIf .
```

```
# loop-back  
SetEnvIf Remote_Addr "127\.0\.0\.1" dontlog  
# robots.txt  
SetEnvIf Request_URI "^/robots\.txt$" dontlog  
#  
CustomLog logs/access_log common env=!dontlog
```

```
SetEnvIf Accept-Language "en" english
CustomLog logs/english_log common env=english
CustomLog logs/non_english_log common env=!english
```



(Log Rotation)

1MB

```
mv access_log access_log.old
mv error_log error_log.old
apachectl graceful
sleep 600
gzip access_log.old error_log.old
```



```
.)
    httpd , userid . ,
root .
. 24 :
```

```
CustomLog "|/usr/local/apache/bin/rotatelogs
/var/log/access_log 86400" common
```

[cronolog](#)



<VirtualHost>

<VirtualHost>      CustomLog ErrorLog

```
LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhost  
CustomLog logs/access_log comonvhost
```

%v

[split-logfile](#)



<a href="#">mod_cgi</a>	<a href="#">PidFile</a>
<a href="#">mod_rewrite</a>	<a href="#">RewriteLog</a>
	<a href="#">RewriteLogLevel</a>
	<a href="#">ScriptLog</a>
	<a href="#">ScriptLogBuffer</a>
	<a href="#">ScriptLogLength</a>

## PID

logs/httpd.pid httpd process id .

[PidFile](#) . process-id

. -k .

[ScriptLog](#)

CGI .

. . [mod\\_cgi](#) .

[mod\\_rewrite](#)

[Rewri](#)

. [RewriteLogLevel](#) .



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

URL

.

.

URL

.



<u>mod alias</u>	<u>Alias</u>
<u>mod proxy</u>	<u>AliasMatch</u>
<u>mod rewrite</u>	<u>CheckSpelling</u>
<u>mod userdir</u>	<u>DocumentRoot</u>
<u>mod spelling</u>	<u>ErrorDocument</u>
<u>mod vhost alias</u>	<u>Options</u>
	<u>ProxyPass</u>
	<u>ProxyPassReverse</u>
	<u>ProxyPassReverseCookieDomain</u>
	<u>ProxyPassReverseCookiePath</u>
	<u>Redirect</u>
	<u>RedirectMatch</u>
	<u>RewriteCond</u>
	<u>RewriteMatch</u>
	<u>ScriptAlias</u>
	<u>ScriptAliasMatch</u>
	<u>UserDir</u>



DOCUMENTROOT

DocumentRoot .

URL-(URL  
DocumentRoot



## DOCUMENTROOT

```
DocumentRoot .  
.  
Options FollowSymLinks  
SymLinksIfOwnerMatch .  
, Alias .
```

```
Alias /docs /var/web
```

```
URL http://www.example.com/docs/dir/file.html  
/var/web/dir/file.html . CGI
```

```
ScriptAlias .
```

```
AliasMatch ScriptAliasMatch
```

```
.,
```

```
ScriptAliasMatch ^/~([a-zA-Z0-9]+)/cgi-bin/(.+) /home/$1/cgi-  
bin/$2
```

```
http://example.com/~user/cgi-bin/script.cgi  
/home/user/cgi-bin/script.cgi, CGI .
```



```
    user          ~user/ .      mod_userdir
, URL
```

```
http://www.example.com/~user/file.html
```

```
    .              Userdir public_html      UserDir
    /etc/passwd    , URL                    /home/u
/home/user/public_html/file.html .
```

```
, Userdir /etc/passwd
```

```
( %7e )      "~"
mod_userdir .
```

```
    . ,              AliasMatch
http://www.example.com/upages/user/file.html
/home/user/public_html/file.html :
```

```
AliasMatch ^/upages/([a-zA-Z0-9]+)/?(.*)
/home/$1/public_html/$2
```



URL , URL  
(redirection) , Redirect . , Document  
/foo/ /bar/ :

```
Redirect permanent /foo/ http://www.example.com/bar/
```

www.example.com /foo/ URL- /foo/ /bar/  
URL .

, RedirectMatch . ,  
:

```
RedirectMatch permanent ^/$  
http://www.example.com/startpage.html
```

```
RedirectMatch temp .*  
http://othersite.example.com/startpage.html
```



(Reverse Proxy)

URL .

(reverse pr

./foo/ , internal.example.com  
/bar/ .

```
ProxyPass /foo/ http://internal.example.com/bar/
ProxyPassReverse /foo/ http://internal.example.com/bar/
```

ProxyPass , ProxyPassReverse  
internal.example.com  
., ProxyPassReverseCookieDomain  
ProxyPassReverseCookieDomain

internal.example.com  
internal.example.com .  
mod\_proxy\_html HTML XHTML .



(Rewriting Engine)

mod\_rewrite .

., mod\_rewrite

(alias), , . mod\_rewrite



## File Not Found

URL  
URL  
"File Not Found" HTML URL  
mod\_speling ( )  
"File Not Found"  
mod\_speling HTTP . ""  
mod\_speling . URL  
mod\_speling URL , ""  
URL  
HTTP status code 404 (file not found)  
ErrorDocument ,



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

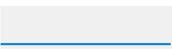
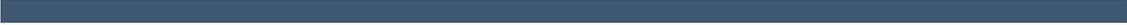
[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Miscellaneous Documentation](#)

--



. , .





. , CGI ,



```
root , User . root
root . root ,
, ServerRoot /usr/local/apache root
```

```
mkdir /usr/local/apache
cd /usr/local/apache
mkdir bin conf logs
chown 0 . bin conf logs
chgrp 0 . bin conf logs
chmod 755 . bin conf logs
```

```
/, /usr, /usr/local root . httpd
:
```

```
cp httpd /usr/local/apache/bin
chown 0 /usr/local/apache/bin/httpd
chgrp 0 /usr/local/apache/bin/httpd
chmod 511 /usr/local/apache/bin/httpd
```

```
htdocs -- root ,
.
root root root .
, httpd . logs (root
root
```



## Server Side Includes (SSI)

```
. SSI SSI
,
, SSI CGI . SSI "exec cmd"
httpd.conf CGI
SSI
SSI CGI
.html .htm SSI .
. SSI .shtml
SSI
IncludesNOEXEC . ScriptAlias Q
#include virtual="..." --> CGI . <
```

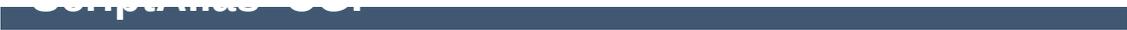


---

CGI / , CGI  
. CGI

CGI ( ) .  
B , B CGI .  
(hook) [suEXEC](#) .  
[CGIWrap](#) .





CGI :

- 
- ,
- , .



CGI . scriptalias  
CGI . , , CGI /

scriptalias CGI .



---

```
mod_php, mod_perl, mod_tcl, mod_python  
( User ),
```

```
· , ·
```



---

.htaccess

.

```
<Directory />  
AllowOverride None  
</Directory>
```

.htaccess .



```
. , URL
```

```
,
```

```
, :
```

```
# cd /; ln -s / public_html  
http://localhost/~root/
```

```
. :
```

```
<Directory />  
Order Deny,Allow  
Deny from all  
</Directory>
```

```
.  
.
```

```
<Directory /usr/users/*/public_html>  
Order Deny,Allow  
Allow from all  
</Directory>  
<Directory /usr/local/httpd>  
Order Deny,Allow  
Allow from all  
</Directory>
```

### Location Directory

```
<Directory /> <Location />
```

UserDir . "/" root

. 1.3 :

```
UserDir disabled root
```



```
grep -c "/jsp/source.jsp?/jsp/ /jsp/source.jsp??" access_log
grep "client denied" error_log | tail -n 10
```

[Source.JSP](#) [Tomcat](#)

10 :

```
[Thu Jul 11 17:18:39 2002] [error] [client foo.bar.com] client
denied by server configuration:
/usr/local/apache/htdocs/.htpasswd
```

.htpasswd

```
foo.bar.com - - [12/Jul/2002:01:59:13 +0200] "GET /.htpasswd
HTTP/1.1"
```

```
<Files ".ht*">
Order allow,deny
Deny from all
</Files>
```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## (DSO)



. DSO , Apache httpd (Dynamic Shared Objects, D  
Extension Tool ([a](#)

.  
DSO .



```
mod_so LoadModule
```

```
mod_so.c DSO .  
DSO . co  
--enable-module=shared DSO .  
mod_foo.so DSO httpd.conf mod_so  
LoadModule .  
( ) DSO apxs (/ eXtenSion) . DSO  
. configure make install C , DSO  
apxs .  
, DSO  
.
```



## Apache 2.2 DSO :

1. `mod_foo.c` DSO  
`mod_foo.so:`

```
$ ./configure --prefix=/path/to/install --enable-foo=shared  
$ make install
```

2. `mod_foo.c` DSO  
`mod_foo.so:`

```
$ ./configure --add-module=module_type:/path/to/3rdparty/mod_foo.c --enable-foo=shared  
$ make install
```

3. :

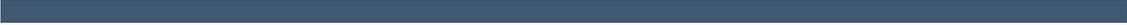
```
$ ./configure --enable-so  
$ make install
```

4. `apxs` m  
`mod_foo.so:`

```
$ cd /path/to/3rdparty  
$ apxs -c mod_foo.c  
$ apxs -i -a -n foo mod_foo.la
```

, `httpd.conf` [LoadModule](#)





(DSO) /(dynamic linking/loading) ,

.

ld.so

dlopen()/dlsym() (loader)

.

DSO (shared libraries) DSO , libfoo.s  
libfoo.so.1.2 . ( /usr/li  
-lfoo .

, LD\_LIBRARY\_PATH /usr/lib  
libfoo.so . ( (unresolved)) (symbol)  
DSO .

DSO (DSO )  
DSO .( )  
libc.so .

DSO (shared objects) DSO ,( foc  
) .  
dlopen() DSO . DSO  
DSO (  
) DSO ( ) . DSO

DSO API dlsym() DSO ,  
(dispatch) . .  
( ) .

DSO , . DSO DSO  
. ? DSO " "(  
, . (

DSO . DSO

.

DSO

.

1998 DSO

(XS DynaLoac

) Perl 5, Netscape Server .

1.3 .

DSO



---

DSO :

- `configure httpd.conf LoadModule`
- `[mod_perl, PHP3]`
- `PHP3, mod_perl, mod_fastcgi`
- `DSO apxs apxs -i apac`  
`restart`

DSO :

- DSO
- 20%
- (position independent code, PIC) (absolute addressing) (relative addressing) 5%
- `DSO ( ld -lfoo) ( ) DSO`  
`ELF a.out ) DSO`  
`. DSO C (`  
`/, ( li`  
`, dlopen() .`



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## (Content Negotiation)

HTTP/1.1 (content negotiation) . media type

''

mod\_negotiation .



media type

```
Accept-Language: fr
```

, , media type  
HTML, media type GIF JPEG

```
Accept-Language: fr; q=1.0, en; q=0.5
Accept: text/html; q=1.0, text/*; q=0.8, image/gif; q=0.6,
image/jpeg; q=0.6, image/*; q=0.5, */*; q=0.1
```

HTTP/1.1 ' (server driven)' .  
Accept - Language, Accept - Charset, Accept - Encoding  
, RFC 2295 RFC 2296  
(transparent)' . RFC ' (feature  
negotiation)' .

**(resource)** (RFC 2396) URI

**(representations)** . media type, ,

( ) .

, **(variant)** .

**(n**  
**(dime**



- `type map ( , *.var ) ,`
- `'MultiViews' .`

## type-map

`type map type-map ( MIME`  
`type application/x-type-map).`

```
AddHandler type-map .var
```

```
Type map ,
HTTP .
, )
. foo.var, foo . map . .(
```

```
URI: foo

URI: foo.en.html
Content-type: text/html
Content-language: en

URI: foo.fr.de.html
Content-type: text/html;charset=iso-8859-2
Content-language: fr, de
```

`typemap , Multiviews` , . media type  
`(JPEG, GIF, ASCII-art)` :  
`(source quality)` :

```
URI: foo

URI: foo.jpeg
Content-type: image/jpeg; qs=0.8

URI: foo.gif
```

```
Content-type: image/gif; qs=0.5
URI: foo.txt
Content-type: text/plain; qs=0.01
```

```
qs 0.000 1.000 . qs 0.000 . 'qs'
1.0 . qs
, JPEG ASCII . ASCII
art ASCII JPEG . qs
```

[mod\\_negotiation typemap](#) .

## Multiviews

```
MultiViews , httpd.conf <Directory>,
<Location>, <Files> ( AllowOverride )
.htaccess Options . Options All
MultiViews . .
```

```
MultiViews : /some/dir/fc
/some/dir/foo MultiViews /some/dir/foo
, foo.* type map . med
type content-encoding .
```

```
MultiViews DirectoryIndex
,
```

```
DirectoryIndex index
```

```
index.html index.html3 .
index.cgi, .
```

```
Charset, Content-Type, Language, Enco
mod_mime , MultiViewsMatch
,, MultiViews .
```



```

type-map
.
.
:
1.
      (quality factor) ".
.
2. (Transparent) RFC 2295
      ' (remote variant      selection algorithm)' .

```

Media	Accept	.
Type	("qs")	.
Language	Accept-Language	. .
	( )	.
Encoding	Accept-Encoding	. .
Charset	Accept-Charset	. .
	media type	.

```

" ()
:
1. ,      Accept* , .
      Accept*      . 4 .
2. " .
      3 .

```

1. Accept media type
2. (language)
3. Accept-Language ()  
LanguagePriority ()
4. (text/html media type ) 'level' media
5. Accept-Charset charset media  
ISO-8859-1 . tex  
type ISO-8859-1
6. ISO-8859-1 charset media . ,
7. user-agent
8. content length .
9. . type-map ,  
ASCII .
3. " . HTTP  
( . ) .
4. ( ) . ("No  
acceptable representation" ) 406  
HTML . , HTML Vary .



## Media Type

Accept: media type . , \*  
"image/\*" /\*/\* " media type . :

```
Accept: image/*, /*/*
```

"image/" type type .  
type . :

```
Accept: text/html, text/plain, image/gif, image/jpeg, /*/*
```

type .

```
Accept: text/html, text/plain, image/gif, image/jpeg, /*/*;  
q=0.01
```

type ( ) 1.0 . /\*/\* 0.01  
type type .

Accept: q /\*/\* , q 0.01  
, "type/\*" (/\*/\* ) 0.02 . Ac  
media type .

## (language)

2.0 .

Accept - langu:  
, "No Acceptable Variant" "Multiple

```

        Accept-language
        ForceLanguagePriority
        LanguagePriority .

, HTTP/1.1          en
        Accept-Language    en-GB
        .)
Acceptable Variants"    LanguagePriority ,
        en-GB en .
        "en-GB; q=0.9, fr; q=0.8" "en" "fr" , "fr"
. HTTP/1.1          ,
        (          URL-) 2.0.47
mod_negotiation prefer-language .
,      mod_negotiation . .

```

```

SetEnvIf Cookie "language=(.+)" prefer-language=$1

```



**(transparent)**

```
(RFC 2295) . {encoding  
..} content-encoding .RVSA/1.0 (RFC 2296)  
, Accept-Encoding  
.RVSA/1.0 5 .
```



(language)

. ( [mod\\_mime](#) .)

MIME-type ( , html), encoding ( , gz),  
( , en) .

:

- foo.en.html
- foo.html.en
- foo.en.html.gz

:

<i>foo.html.en</i>	foo foo.html	-
<i>foo.en.html</i>	foo	foo.html
<i>foo.html.en.gz</i>	foo foo.html	foo.gz foo.html.gz
<i>foo.en.html.gz</i>	foo	foo.html foo.html.gz foo.gz
<i>foo.gz.html.en</i>	foo foo.gz foo.gz.html	foo.html
<i>foo.html.gz.en</i>	foo foo.html foo.html.gz	foo.gz

( , foo)

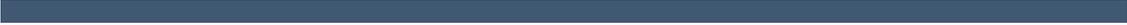
,

html

.

MIME-type ( , foo.html) (encoding  
) MIME-type ( , foo.html.en)





URL .

URL .

HTTP/1.1

HTTP/1.0 . ,

CacheNegotiatedDocs HTTP/1.0 ( )

HTTP/1.1 .

HTTP/1.1

Vary HTTP .



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



"500 Server Error" ( )  
) URL .



## NCSA httpd 1.3

:

1. NCSA
2. URL
3. URL .

URL ,

CGI

:

```
REDIRECT_HTTP_ACCEPT=*/, image/gif, image/x-xbitmap,
image/jpeg
REDIRECT_HTTP_USER_AGENT=Mozilla/1.1b2 (X11; I; HP-UX A.09.05
9000/712)
REDIRECT_PATH=./bin:/usr/local/bin:/etc
REDIRECT_QUERY_STRING=
REDIRECT_REMOTE_ADDR=121.345.78.123
REDIRECT_REMOTE_HOST=ooh.ahhh.com
REDIRECT_SERVER_NAME=crash.bang.edu
REDIRECT_SERVER_PORT=80
REDIRECT_SERVER_SOFTWARE=Apache/0.8.15
REDIRECT_URL=/cgi-bin/buggy.pl
```

REDIRECT\_ .

```
REDIRECT_URL REDIRECT_QUERY_STRING (cgi-script cgi-
include) URL . ( REDIR
) . ErrorDocument ( http: (scheme)
)
```



[AllowOverride](#) .htaccess

[ErrorDocument](#)

...

```
ErrorDocument 500 /cgi-bin/crash-recover
ErrorDocument 500 "Sorry, our script crashed. Oh dear"
ErrorDocument 500 http://xxx/
ErrorDocument 404 /Lame_excuses/not_found.html
ErrorDocument 401 /Subscription/how_to_subscribe.html
```

,

```
ErrorDocument <3-digit-code> <action>
```

action,

1. . (") . . : (
2. URL.
3. URL.



---

URL /server-include .

CGI . .

CGI REDIRECT\_ . REDIRECT\_  
REDIRECT\_HTTP\_USER\_AGENT . , HTTP\_USER\_AGE  
REDIRECT\_URL REDIRECT\_STATUS . URL URL

ErrorDocument CGI ,  
"Status:" . , Perl ErrorDocument  
:

```
...  
print "Content-type: text/html\n";  
printf "Status: %s Condition Intercepted\n",  
$ENV{"REDIRECT_STATUS"};  
...
```

404 Not Found , ( ; )

( ) Location: ,  
Status: . Location: .



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## (Binding)



[DNS](#)



```
core <VirtualHost>
mpm_common Listen
```

IP,

Listen

List

Listen

, 80 8000

:

```
Listen 80
Listen 8000
```

```
Listen 192.0.2.1:80
Listen 192.0.2.5:8000
```

IPv6 :

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```



IPv6 APR IPv6 , IPv6  
IPv6 .

IPv6 IPv4 IPv6 .  
IPv4-(mapped) IPv6 IPv6 IPv4 , FreeBSD  
NetBSD OpenBSD .

Tru64 IPv4 IPv6  
IPv4 IPv6 , IPv4- IPv6  
enable-v4-mapped .

--enable-v4-mapped FreeBSD, NetBSD, OpenBSD

APR IPv4 ,  
:

```
Listen 0.0.0.0:80  
Listen 192.0.2.1:80
```

IPv4 IPv6 (IPv4-  
[configure](#) --disable-v4-mapped . --disable-v4-  
mapped FreeBSD, NetBSD, OpenBSD .



```
Listen .
<VirtualHost> , .
<VirtualHost> .
.
. <VirtualHost> .
```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

**(MPM)**

.

(Multi-Processing Module) ,

.



Apache 2.0

(Multi-Processing Modu

- mpm\_winnt Apache 1.3

POSIX

MPM

- worker MPM ,  
preforking MPM .  
(perchild)

(scalability)

MPM

MPM

MPM



---

MPMs .

MPM , MPM

MPM ./configure with-mpm= *NAME* . *N*  
MPM .

./httpd -l MPM . MPM

.



---

MPM . MPM .

BeOS	<a href="#"><u>beos</u></a>
Netware	<a href="#"><u>mpm_netware</u></a>
OS/2	<a href="#"><u>mpmt_os2</u></a>
	<a href="#"><u>prefork</u></a>
	<a href="#"><u>mpm_winnt</u></a>



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



*(environment variable)* .  
., CGI .  
.

,  
Side Include .  
.



<u>mod_env</u>	<u>BrowserMatch</u>
<u>mod_rewrite</u>	<u>BrowserMatchNoCase</u>
<u>mod_setenvif</u>	<u>PassEnv</u>
<u>mod_unique_id</u>	<u>RewriteRule</u>
	<u>SetEnv</u>
	<u>SetEnvIf</u>
	<u>SetEnvIfNoCase</u>
	<u>UnsetEnv</u>

SetEnv .

, mod\_setenvif . ,  
 (User-Agent) Referer ( )  
 mod\_rewrite RewriteRule [E=...]

mod\_unique\_id "" ()  
 UNIQUE\_ID .

## CGI

CGI SSI

- CGI
- [suexec](#) CGI , CGI  
suexec.c .
- , , , ,  
. CGI SSI .



<u>mod authz host</u>	<u>Allow</u>
<u>mod cgi</u>	<u>CustomLog</u>
<u>mod ext filter</u>	<u>Deny</u>
<u>mod headers</u>	<u>ExtFilterDefine</u>
<u>mod include</u>	<u>Header</u>
<u>mod log config</u>	<u>LogFormat</u>
<u>mod rewrite</u>	<u>RewriteCond</u>
	<u>RewriteRule</u>

**CGI**

CGI .  
 CGI . [CGI](#) .

**SSI**

mod\_include INCLUDES (SSI) echo  
 ,  
 CGI . [SSI](#) .

allow from env= deny from env=  
 . [SetEnvIf](#)  
 , (User-Agent) .

[LogFormat](#) %e . , [Cusi](#)  
 . [SetEnv](#)  
 . , gif ,  
 .



---

`SetEnv PassEnv .`

## **downgrade-1.0**

`HTTP/1.0`

## **force-gzip**

`DEFLATE accept-encoding`

## **force-no-vary**

`Vary .`

`force-response-1.0 .`

## **force-response-1.0**

`HTTP/1.0 HTTP/1.0`

`. AOL .`

`HTTP/1.0 HTTP/1.1`

`, .`

## **gzip-only-text/html**

`"1" text/html content-type`

`mod\_deflate`

`DEFLATE . (gzip "identity"`

`)`

`mod\_negotiation .`

## **no-gzip**

`mod\_deflate DEFLATE ,`

`mod\_negotiation .`

## **nokeepalive**

`KeepAlive .`

## prefer-language

mod\_negotiation . ( en, ja, x-kling  
, mod\_negotiation .  
.

## redirect-carefully

WebFolders DAV

## suppress-error-charset

2.0.40

ISO-8859-1 (

· , ·

· ,



## httpd.conf

```
#
#   HTTP
#   Netscape 2.x
#   keepalive . . .
#   HTTP/1.1 301 302
#   () keepalive
#   Microsoft Internet Explorer 4.0b2
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

#
#   HTTP/1.1
#   HTTP/1.0 HTTP/1.1
#
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
```

```
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
```

""

## /web/images

```
SetEnvIf Referer "^http://www.example.com/" local_referal
# Referer
SetEnvIf Referer "^$" local_referal
```

```
<Directory /web/images>
  Order Deny,Allow
  Deny from all
  Allow from env=local_referal
</Directory>
```

ApacheToday " [Keeping Your Images from Adorning Other Sites](#)" .



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

--



.





---

## CGI

html footer.pl CGI .

```
Action add-footer /cgi-bin/footer.pl
AddHandler add-footer .html
```

CGI (PATH\_TRANSLATED ) .

## HTTP

HTTP send-as-is .  
/web/htdocs/asis/ send

```
<Directory /web/htdocs/asis>
SetHandler send-as-is
</Directory>
```



[Apache API](#) .

request

```
char *handler
```

```
, invoke_handler r->handle
```

```
. content type .
```

```
type .
```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

--



.



<u>mod deflate</u>	<u>AddInputFilter</u>
<u>mod ext filter</u>	<u>AddOutputFilter</u>
<u>mod include</u>	<u>RemoveInputFilter</u>
	<u>RemoveOutputFilter</u>
	<u>ExtFilterDefine</u>
	<u>ExtFilterOptions</u>
	<u>SetInputFilter</u>
	<u>SetOutputFilter</u>

*(filter)*

*(output filter)* . ,

*(byte-range)* . ,

. SetInputFilter, SetOutputFilter, AddInputFilter,  
AddOutputFilter, RemoveInputFilter,  
RemoveOutputFilter .

## INCLUDES

mod include Server-Side Includes

## DEFLATE

mod deflate

, mod ext filter .



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# suEXEC

suEXEC CGI SSI ID ID .  
CGI SSI .  
CGI SSI  
suEXEC .  
suEXEC .



---

setuid setgid  
suEXEC

, setuid

, suEXEC suEXE

, suEXEC  
. suEXEC suEXEC

suEXEC

?? . !



suEXEC

. suEXEC

**suEXEC** setuid

"wrapper" . wrapper

userid CGI SSI

HTTP .

suEXEC wrapper

wrapper .

. :

1. **wrapper**

?

wrapper

2. **wrapper ?**

wrapper .

. wrapper

suEXEC

3. **wrapper ?**

wrapper ?

()

4. **CGI SSI**

?

CGI SSI '/'

'..? .

CGI/SSI suEXEC root (  
docroot=DIR)

--with-suexec-

5. ?

?

6. ?

?

7. **superuser** ?

suEXEC root CGI/SSI .

8. **userid ID** ?

ID . CGI/SSI userid  
."'" .

9. **superuser** ?

suEXEC root CGI/SSI .

10. **groupid ID** ?

ID . CGI/SSI groupid  
."'" .

11. **wrapper** ?

setuid setgid . ,

12. **CGI/SSI** ?

13. ?

suEXEC root  
UserDir suEXEC userdir (  
 ) ?

14. ?

15. **CGI/SSI ?**

16. **CGI/SSI ?**

CGI/SSI .

17. **CGI/SSI setuid setgid ?**

UID/GID .

18.  **/ / ?**

?

19. ?

suEXEC ( ) PATH , ( )

20. **CGI/SSI ?**

suEXEC CGI/SSI .

suEXEC wrapper . CGI/SSI ,

suEXEC

" " .



## suEXEC

### --enable-suexec

```
suEXEC . APACI suEX  
enable-suexec --with-suexec-xxxxx
```

### --with-suexec-bin=PATH

```
suexec .  
with-suexec-bin=/usr/sbin/suexec
```

### --with-suexec-caller=UID

### --with-suexec-userdir=DIR

```
suEXEC .  
, "" . ( , "*" ) "" UserDir  
. UserDir passwd suEXEC  
. "public_html".  
UserDir ,  
. , "~userdir" cgi !
```

### --with-suexec-docroot=DIR

```
DocumentRoot . suEXEC (UserDirs  
) . --datadir "/htdocs" .  
-datadir="/home/apache" suEXEC wrapper  
document root "/home/apache/htdocs" .
```

### --with-suexec-uidmin=UID

```
suEXEC UID . 500 100 .  
100.
```

### --with-suexec-gidmin=GID

```
suEXEC GID . 100 .
```

### --with-suexec-logfile=FILE

```
suEXEC ( ) .
```

```
"suexec_log"                                (--logfiledir).  
--with-suexec-safepath=PATH  
CGI_PATH .  
"/usr/local/bin:/usr/bin:/bin".
```

### suEXEC wrapper

```
--enable-suexec suEXEC                       make s  
( ) .  
make install .  
"/usr/local/apache2/sbin/suexec".  
root . wrapper ID  
setuserid .
```

```
suEXEC wrapper                               --with-suexec-ca  
, suEXEC  
suEXEC  
, :
```

```
User www  
Group webgroup
```

```
suexec "/usr/local/apache2/sbin/suexec" , :
```

```
chgrp webgroup /usr/local/apache2/bin/suexec  
chmod 4750 /usr/local/apache2/bin/suexec
```

```
suEXEC wrapper .
```



```
    --sbindir                suexec (
"/usr/local/apache2/sbin/suexec") .    suEXEC wrapper
(error                          log) :
```

```
[notice] suEXEC mechanism enabled (wrapper: /path/to/suexec)
```

```
    wrapper ,
```

```
    .
    suEXEC
    USR1 .
    suEXEC suexec .
```



CGI SuexecUserGroup  
mod\_userdir suEXEC wrapper .

:  
suEXEC wrapper VirtualHost  
SuexecUserGroup . ID  
<VirtualHost> User Group .  
userid .

:  
mod\_userdir suEXEC wrapper ,  
ID CGI . ID CGI  
.  
--with-suexec-userdir .



suEXEC wrapper  
. wrapper

--with-suexec-logfile  
err



! . suEXEC ""  
wrapper .

- suEXEC
- 

suEXEC document r  
userdir document root .  
suEXEC document root  
(.)

- suEXEC PATH

- suEXEC



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Miscellaneous Documentation](#)

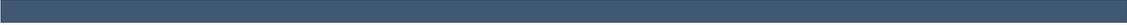


2.0

1.3 2.0 (scalability)

2.0





""

Maxi

top

: CPU, ,

, ""

:

TCP

- sendfile(2) ,  
Solaris 8 .)
- CPU



<u>mod_dir</u>	<u>AllowOverride</u>
<u>mpm_common</u>	<u>DirectoryIndex</u>
<u>mod_status</u>	<u>HostnameLookups</u>
	<u>EnableMMAP</u>
	<u>EnableSendfile</u>
	<u>KeepAliveTimeout</u>
	<u>MaxSpareServers</u>
	<u>MinSpareServers</u>
	<u>Options</u>
	<u>StartServers</u>

## HostnameLookups DNS

```
1.3 HostnameLookups On. DNS
. 1.3 Off.
logresolve .
```

```
Allow from domain Deny from domain (, IP
) - DNS (
IP .
```

```
<Location /server-status>
DNS . .html .cgi DNS
```

```
HostnameLookups off
<Files ~ "\.(html|cgi)$">
  HostnameLookups on
</Files>
```

CGI DNS ,

CGI gethostbyi

## FollowSymLinks SymLinksIfOwnerMatch

URL Options FollowSymLinks Options  
SymLinksIfOwnerMatch

. , :

```
DocumentRoot /www/htdocs  
<Directory />  
  Options SymLinksIfOwnerMatch  
</Directory>
```

/index.html URI . /www,  
/www/htdocs, /www/htdocs/index.html lstat(2)  
. lstats .  
:

```
DocumentRoot /www/htdocs  
<Directory />  
  Options FollowSymLinks  
</Directory>  
  
<Directory /www/htdocs>  
  Options -FollowSymLinks +SymLinksIfOwnerMatch  
</Directory>
```

DocumentRoot . DocumentRoot  
Alias RewriteRule .  
, FollowSymLinks, SymLinksIfOwnerMatch  
.

## AllowOverride

URL overrides ( .htaccess )  
.htaccess . ,

```
DocumentRoot /www/htdocs
```

```
<Directory />
  AllowOverride all
</Directory>
```

```
/index.html URI . / .htaccess,
/www/.htaccess, /www/htdocs/.htaccess .
Options FollowSymLinks .
AllowOverride None .
```

:

```
DirectoryIndex index
```

:

```
DirectoryIndex index.cgi index.pl index.shtml index.html
```

```
, MultiViews, ty
map .
```

```
Options MultiViews type-map
. type-map .
```

### **(memory-mapping)**

```
, server-side-include 2.0
mmap(2) .
```

- mmap CPU read(2) . ,

Solaris 2.0

mmap

- NFS

NFS

bus error

EnableMMAP off

.)

## Sendfile

sendfile(2)

sendfile -- ,

sendfile read send

. sendfile

:

- sendfile
- sendfile

- NFS

sendfile

EnableSendfile off

(: .)

### 1.3 MinSpareServers, MaxSpareServers, StartServers

"" . StartServers , MinSpare  
StartServers 5 100  
 95 . , 10

. 1.3

, 1 ,

, 1 , ,

3.

MinSpareServers .

MinSpareServers, MaxSpareServers,  
StartServers . 4 Er  
mod\_status .

MaxRequestsPerChild .  
0. 30 , . SunO!  
Solaris , 10000 .

(keep-alive)  
KeepAliveTimeout 15 .  
60



---

## MPM

2.x (MPMs) . MPM .  
[beos](#), [mpm\\_netware](#), [mpmt\\_os2](#), [mpm\\_winnt](#)  
MPM . MPM .  
(scalability) MPM :

- [worker](#) MPM .  
worker prefork MPM .
- [prefork](#) MPM .  
prefork worker , .  
prefork worker : (thread-safe)  
,

MPM MPM MPM .

### [LoadModule](#) .

[mod\\_dir](#), [mod\\_log\\_config](#) .  
[mod\\_log\\_config](#) .

## Atomic

[mod\\_cache](#) worker MPM APR atomic API .  
API atomic .

APR /CPU . ,  
CPU atomic compare-and-swap (CAS) .  
APR CPU mutex

CPU ,  
atomics atomic :

```
./buildconf  
./configure --with-mpm=worker --enable-nonportable-atomics=yes
```

--enable-nonportable-atomics :

- SPARC Solaris  
APR Solaris/SPARC mutex atomic .  
enable-nonportable-atomics APR  
compare-and-swap SPARC v8plus .  
atomic (CPU ),  
UltraSPARC .
- Linux on x86  
APR mutex atomic .  
nonportable-atomics APR compare-and-  
swap 486 . atomic ,  
(386) .

## mod\_status ExtendedStatus On

mod\_status ExtendedStatus On  
gettimeofday(2)( times(2)) (1.3)  
time(2) .  
ExtendedStatus off .

## accept -

```
:  
2.0 . ,  
.
```



. (

```
for (;;) {
    accept_mutex_on ();
    for (;;) {
        fd_set accept_fds;

        FD_ZERO (&accept_fds);
        for (i = first_socket; i <= last_socket; ++i) {
            FD_SET (i, &accept_fds);
        }
        rc = select (last_socket+1, &accept_fds, NULL, NULL,
                    NULL);
        if (rc < 1) continue;
        new_connection = -1;
        for (i = first_socket; i <= last_socket; ++i) {
            if (FD_ISSET (i, &accept_fds)) {
                new_connection = accept (i, NULL, NULL);
                if (new_connection != -1) break;
            }
        }
        if (new_connection != -1) break;
    }
    accept_mutex_off ();
    process the new_connection;
}
```

accept\_mutex\_on accept\_mutex\_off mutex .  
mutex . mutex . (1.3  
src/conf.h (1.3 ) src/include/ap\_config.h  
(locking) ,

[AcceptMutex](#) mutex .

### AcceptMutex flock

flock(2) ( [LockFile](#)

### AcceptMutex fcntl

fcntl(2) ( [LockFile](#)

### AcceptMutex sysvsem

(1.3 ) SysV mutex . SysV

.  
(  
). uid CGI ( , suexec cgiw  
CGI) API  
( IRIX ).

### AcceptMutex pthread

(1.3 ) POSIX mutex POSIX  
, (2.5 ) Solaris .

### AcceptMutex posixsem

(2.0 ) POSIX . mutex  
(segfault) .

(serialization) APR .

### Listen

### accept -

, ?  
, . (non-blocking)  
"(spinning)" . TCP

. ( 2.0.30, 128Mb Pentium pro)

3% . 100ms

. LAN .

SINGLE\_LISTEN\_UNSERIALIZED\_ACCEPT .

## Close (lingering)

[draft-ietf-http-connection-00.txt](#) 8

(TCP , ).

.

.TCP

, . 1.2

. TCP/IP

( , SunOS4 -- )

.

. SO\_LINGER . TCP/IP

. ( , 2.0.31)

.

( http\_main.c ) lingering\_close .

:

```
void lingering_close (int s)
{
    char junk_buffer[2048];

    /* shutdown the sending side */
    shutdown (s, 1);

    signal (SIGALRM, lingering_death);
    alarm (30);

    for (;;) {
        select (s for reading, 2 second timeout);
        if (error) break;
        if (s is ready for reading) {
            if (read (s, junk_buffer, sizeof (junk_buffer)) <= 0) {
                break;
            }
            /* just toss away whatever is here */
        }
    }

    close (s);
}
```

```

CPU , . HTTP/1.1
(persistent),
, . HTTP/1.1
) lingering_close (
).

```

## Scoreboard

```

scoreboard . scoreboard
.
. ( ).
src/main/conf.h USE_MMAP_SCOREBOARD
USE_SHMGET_SCOREBOARD . ( HAVE_SHMGET
HAVE_SHMGET )
src/main/http_main.c (hook) .
( .)

```

```

: 1.2
.

```

## DYNAMIC\_MODULE\_LIMIT

```

(
-DDYNAMIC_MODULE_LIMIT=0 .

```



Solaris 8 worker MPM 2.0.38

(trace).

:

```
truss -l -p httpd_child_pid.
```

-l truss LWP (lightweight process, --Solaris) ID .

strace, ktrace, par . .

10KB .  
( ).

```
/67: accept(3, 0x00200BEC, 0x00200C0C, 1) (sleeping...)
/67: accept(3, 0x00200BEC, 0x00200C0C, 1) = 9
```

(listener) LWP #67 .

```
accept(2) . worker MPM
accept .
```

```
/65: lwp_park(0x00000000, 0) = 0
/67: lwp_unpark(65, 1) = 0
```

(accept) worker .  
worker LWP #65 .

```
/65: getsockname(9, 0x00200BA4, 0x00200BC4, 1) = 0
```

(local) . (  
) .

```
/65: brk(0x002170E8) = 0
/65: brk(0x002190E8) = 0
```

brk(2) (heap) .  
apr\_bucket\_alloc)

malloc(3) .

```
/65: fcntl(9, F_GETFL, 0x00000000) = 2  
/65: fstat64(9, 0xFAF7B818) = 0  
/65: getsockopt(9, 65535, 8192, 0xFAF7B918, 0xFAF7B910, 219065) = 0  
/65: fstat64(9, 0xFAF7B818) = 0  
/65: getsockopt(9, 65535, 8192, 0xFAF7B918, 0xFAF7B914, 219065) = 0  
/65: setsockopt(9, 65535, 8192, 0xFAF7B918, 4, 2190656) = 0  
/65: fcntl(9, F_SETFL, 0x00000082) = 0
```

worker (9) (non-blocking) .  
setsockopt(2) getsockopt(2) Solaris libc  
fcntl(2) .

```
/65: read(9, " G E T / 1 0 k . h t m" . . , 8000) = 97
```

worker .

```
/65: stat("/var/httpd/apache/httpd-8999/htdocs/10k.html", 0xFAF7B818) = 0  
/65: open("/var/httpd/apache/httpd-8999/htdocs/10k.html", O_RDONLY) = 9
```

Options FollowSymLinks AllowOverride None.

lstat(2) .htaccess .

1) , 2) , stat(2) .

```
/65: sendfilev(0, 9, 0x00200F90, 2, 0xFAF7B53C) = 10269
```

sendfilev(2) HTTP .

Sendfile . sendfile(2)

write(2) writev(2) .

```
/65: write(4, " 1 2 7 . 0 . 0 . 1 - " . . , 78) = 78
```

```

write(2) (access log) . time(2
1.3 2.0 gettimeofday(3) .
gettimeofday Solaris .

```

```

/65: shutdown(9, 1, 1) = 0
/65: poll(0xFAF7B980, 1, 2000) = 1
/65: read(9, 0xFAF7BC20, 512) = 0
/65: close(9) = 0

```

worker (lingering close).

```

/65: close(10) = 0
/65: lwp_park(0x00000000, 0) (sleeping...)

```

worker , (listener)

```

/67: accept(3, 0x001FEB74, 0x001FEB94, 1) (sleeping...)

```

```

(worker worker MPM
) worker . , worker
accept(2) ( ) .

```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >



IP

ServerPath



IP IP

HTTP .

. IP

.

DNS

IP ,

.

IP .

. IP :

•

HTTP/1.1 , HTTP/1.0

• SSL SSL

• IP

(bandwidth)



```

core DocumentRoot
      NameVirtualHost
      ServerAlias
      ServerName
      ServerPath
      <VirtualHost>

```

```

NameVirtualHost . IP ( ) .
NameVirtualHost * . ( , SSL )
*:80 . NameVirtualHost IP
IP .
.
    <VirtualHost> . <VirtualHost>>
    NameVirtualHost ( , IP
<VirtualHost>> ServerName
    DocumentRoot .

```

```

ServerName DocumentRoot ServerName
DocumentRoot .

```

```

www.domain.tld IP
www.otherdomain.tld . httpd.
:

```

```

NameVirtualHost *:80
<VirtualHost *:80>

```

```

    ServerName www.domain.tld
    ServerAlias domain.tld *.domain.tld
    DocumentRoot /www/domain
</VirtualHost>

<VirtualHost *:80>
    ServerName www.otherdomain.tld
    DocumentRoot /www/otherdomain
</VirtualHost>

```

NameVirtualHost <VirtualHost>

\* IP

. , IP

, IP

.

.

<VirtualHost>

ServerAlias .

<Virtual

ServerAlias :

```

ServerAlias domain.tld *.domain.tld

```

domain.tld

www.domain.tld .

\* ? .

ServerName ServerAl:

.

IP DNS .

<<VirtualHost>>

,

)

NameVirtualHost IP . IP

<VirtualHost>

ServerName

. . , IP

. IP

NameVirtualHost ,

DocumentRoot .



( )

?

Host .

ServerPath :

:

```
NameVirtualHost 111.22.33.44
<VirtualHost 111.22.33.44>
  ServerName www.domain.tld
  ServerPath /domain
  DocumentRoot /web/domain
</VirtualHost>
```

```
? " /domain" URI www.domain.
. , Host: http://www.domain.tld/ ,
http://www.domain.tld/domain/ .
http://www.domain.tld/do
. (, "file.html"
"./icons/image.gif")
("http://www.domain.tld/domain/misc/file.html"
"/domain/misc/file.html") /domain/ .
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

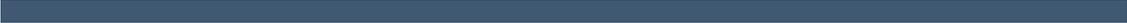
IP



---

```
IP          IP IP
              ( . "ip aliases"
"ifconfig" ) .
```





```
.  
.  
:  
• 2 1  
      User, Group, Listen, ServerRoot  
.  
• , IP (file descriptor) . ""  
  Listen . , (  
              )  
:  
• .  
• .
```



. [Listen](#) IP ( ).

Listen www.smallco.com:80

IP . ( [DNS](#) )



[ServerAdmin](#), [ServerName](#), [DocumentRoot](#), [ErrorLog](#),  
[TransferLog](#), [CustomLog](#) . ,

```
<VirtualHost www.smallco.com>
ServerAdmin webmaster@mail.smallco.com
DocumentRoot /groups/smallco/www
ServerName www.smallco.com
ErrorLog /groups/smallco/logs/error_log
TransferLog /groups/smallco/logs/access_log
</VirtualHost>

<VirtualHost www.baygroup.org>
ServerAdmin webmaster@mail.baygroup.org
DocumentRoot /groups/baygroup/www
ServerName www.baygroup.org
ErrorLog /groups/baygroup/logs/error_log
TransferLog /groups/baygroup/logs/access_log
</VirtualHost>
```

IP . ([DNS](#) )

VirtualHost  
VirtualHost

[suEXEC](#) VirtualHost [User](#) [Group](#) .

:  
.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >



1.3



httpd.conf

<VirtualHost>

:

```
NameVirtualHost 111.22.33.44
<VirtualHost 111.22.33.44>
    ServerName www.customer-1.com
    DocumentRoot /www/hosts/www.customer-1.com/docs
    ScriptAlias /cgi-bin/ /www/hosts/www.customer-1.com/cgi-bin
</VirtualHost>
<VirtualHost 111.22.33.44>
    ServerName www.customer-2.com
    DocumentRoot /www/hosts/www.customer-2.com/docs
    ScriptAlias /cgi-bin/ /www/hosts/www.customer-2.com/cgi-bin
</VirtualHost>
#
<VirtualHost 111.22.33.44>
    ServerName www.customer-N.com
    DocumentRoot /www/hosts/www.customer-N.com/docs
    ScriptAlias /cgi-bin/ /www/hosts/www.customer-N.com/cgi-bin
</VirtualHost>
```

<VirtualHost> .

1.

.

2.

DNS . ,

.  
) .  
fifo ,



```

IP HTTP      Host: .
.
.           mod_vhost_alias , 1.3.6
.           mod_rewrite .
.
.
.           \'.
.           ServerName , CGI           SERVER_NAME .
UseCanonicalName .           UseCanonicalName Off
Host: .           UseCanonicalName DNS IP DNS
.           , IP
.           ServerName .
.
. ( DocumentRoot,           CGI DOCUMENT_ROOT )
. core           URI ,
. ( mod_vhost_alias mod_rewrite) .
.           DOCUMENT_ROOT           CGI SSI
.

```



## mod\_vhost\_alias

```
# Host:
UseCanonicalName Off

#
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

#
VirtualDocumentRoot /www/hosts/%0/docs
VirtualScriptAlias /www/hosts/%0/cgi-bin
```

```
UseCanonicalName Off UseCanonicalName DNS
IP . IP .
```



---

```
ISP .
www.user.isp.com /home/user/
. cgi-bin .
```

```
# .
#
VirtualDocumentRoot /www/hosts/%2/docs
# cgi-bin
ScriptAlias /cgi-bin/ /www/std-cgi/
```

mod\_vhost\_alias

VirtualDocumentRoot

.



<VirtualHost>

.. , IP , IP .  
<VirtualHost> .

```
UseCanonicalName Off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory /www/commercial>
  Options FollowSymLinks
  AllowOverride All
</Directory>

<Directory /www/homepages>
  Options FollowSymLinks
  AllowOverride None
</Directory>

<VirtualHost 111.22.33.44>
  ServerName www.commercial.isp.com

  CustomLog logs/access_log.commercial vcommon

  VirtualDocumentRoot /www/commercial/%0/docs
  VirtualScriptAlias /www/commercial/%0/cgi-bin
</VirtualHost>

<VirtualHost 111.22.33.45>
  ServerName www.homepages.isp.com

  CustomLog logs/access_log.homepages vcommon

  VirtualDocumentRoot /www/homepages/%0/docs
  ScriptAlias /cgi-bin/ /www/std-cgi/
</VirtualHost>
```



IP . DN  
IP .  
, DNS .

```
# IP DNS
UseCanonicalName DNS

# IP
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# IP
VirtualDocumentRootIP /www/hosts/%0/docs
VirtualScriptAliasIP /www/hosts/%0/cgi-bin
```



```
1.3.6 mod_vhost_alias .
mod_vhost_alias mod_rewrite
Host: , .

. 1.3.6 %V , 1.3.0
%v . 1.3.4 .
.htaccess UseCanonicalName
. %{Host}i Host:
. , %V :port .
```







```
RewriteEngine on

RewriteMap lowercase int:tolower

# CGI
RewriteCond %{REQUEST_URI} !^/cgi-bin/

# RewriteRule
RewriteCond ${lowercase:%{SERVER_NAME}} ^www\.[a-z-
]+\.isp\.com$

# URI
# [C]
RewriteRule ^(.+) ${lowercase:%{SERVER_NAME}}$1 [C]

#
RewriteRule ^www\.[a-z-]+\\.isp\.com/(.*) /home/$1/$2

# CGI
ScriptAlias /cgi-bin/ /www/std-cgi/
```



```
mod_rewrite
```

```
vhost.map :
```

```
www.customer-1.com /www/customers/1
www.customer-2.com /www/customers/2
# ...
www.customer-N.com /www/customers/N
```

```
http.conf :
```

```
RewriteEngine on

RewriteMap lowercase int:tolower

#
RewriteMap vhost txt:/www/conf/vhost.map

# alias
RewriteCond %{REQUEST_URI} !^/icons/
RewriteCond %{REQUEST_URI} !^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
#
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/docs/$1

RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/cgi-bin/$1
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

[Empty rectangular box]



IP



IP , DNS (CNAMEs) .  
www.example.com www.example.org .

## Note

DNS . hosts  
hosts .

```
# 80
Listen 80

# IP
NameVirtualHost *:80

<VirtualHost *:80>
  DocumentRoot /www/example1
  ServerName www.example.com
  #
</VirtualHost>

<VirtualHost *:80>
  DocumentRoot /www/example2
  ServerName www.example.org
  #
</VirtualHost>
```

, . www.exam  
, . ServerName  
VirtualHost .

\* IP . VirtualHost

NameVirtualHost

:

```
NameVirtualHost 172.20.30.40  
<VirtualHost 172.20.30.40>  
# ...
```

ISP IP  
, IP

IP

\*

IP



IP .

IP . (172.20.30.40) "" server.domain  
, (172.20.30.50) .

```
Listen 80

# 172.20.30.40 ""
ServerName server.domain.com
DocumentRoot /www/mainserver

#
NameVirtualHost 172.20.30.50

<VirtualHost 172.20.30.50>
  DocumentRoot /www/example1
  ServerName www.example.com

  # ...
</VirtualHost>

<VirtualHost 172.20.30.50>
  DocumentRoot /www/example2
  ServerName www.example.org

  # ...
</VirtualHost>
```

172.20.30.50 . ,  
172.20.30.50 www.example.com .



```
IP ( 192.168.1.1 172.20.30.40) . (
) 0 . server.examp
(172.20.30.40), ( 192.168
```

VirtualHost .

```
NameVirtualHost 192.168.1.1
NameVirtualHost 172.20.30.40

<VirtualHost 192.168.1.1 172.20.30.40>
  DocumentRoot /www/server1
  ServerName server.example.com
  ServerAlias server
</VirtualHost>
```

VirtualHost .

```
:
server.example.com server
IP * .
```



IP . "NameVirtualHost"  
. NameVirtualHost name:port <VirtualHost name:port>  
Listen .

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
  ServerName www.example.com
  DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  ServerName www.example.com
  DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
  ServerName www.example.org
  DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  ServerName www.example.org
  DocumentRoot /www/otherdomain-8080
</VirtualHost>
```



```
www.example.com www.example.org IP  
(172.20.30.40 172.20.30.50).
```

```
Listen 80  
  
<VirtualHost 172.20.30.40>  
  DocumentRoot /www/example1  
  ServerName www.example.com  
</VirtualHost>  
  
<VirtualHost 172.20.30.50>  
  DocumentRoot /www/example2  
  ServerName www.example.org  
</VirtualHost>
```

```
<VirtualHost>           (,           loca
```

```
.
```



www.example.com www.example.org IP  
(172.20.30.40 172.20.30.50) . IP 80 8080

```
Listen 172.20.30.40:80
Listen 172.20.30.40:8080
Listen 172.20.30.50:80
Listen 172.20.30.50:8080

<VirtualHost 172.20.30.40:80>
  DocumentRoot /www/example1-80
  ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  DocumentRoot /www/example1-8080
  ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.50:80>
  DocumentRoot /www/example2-80
  ServerName www.example.org
</VirtualHost>

<VirtualHost 172.20.30.50:8080>
  DocumentRoot /www/example2-8080
  ServerName www.example.org
</VirtualHost>
```



, IP

```
Listen 80

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
    DocumentRoot /www/example1
    ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.40>
    DocumentRoot /www/example2
    ServerName www.example.org
</VirtualHost>

<VirtualHost 172.20.30.40>
    DocumentRoot /www/example3
    ServerName www.example3.net
</VirtualHost>

# IP-
<VirtualHost 172.20.30.50>
    DocumentRoot /www/example4
    ServerName www.example4.edu
</VirtualHost>

<VirtualHost 172.20.30.60>
    DocumentRoot /www/example5
    ServerName www.example5.gov
</VirtualHost>
```



## \_default\_

IP .

```
<VirtualHost _default_:*>
  DocumentRoot /www/default
</VirtualHost>
```

default()

default

( / ).

AliasMatch RewriteRule ()

.

## \_default\_

, 80

\_defau

```
<VirtualHost _default_:80>
  DocumentRoot /www/default80
  # ...
</VirtualHost>

<VirtualHost _default_:*>
  DocumentRoot /www/default
  # ...
</VirtualHost>
```

80 default ( )

.

## \_default\_

80 default .

```
<VirtualHost _default_:80>  
DocumentRoot /www/default  
...  
</VirtualHost>
```

80

,





HTTP/1.0

).

.

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
  # primary vhost
  DocumentRoot /www/subdomain
  RewriteEngine On
  RewriteRule ^/.*/www/subdomain/index.html
  # ...
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/subdomain/sub1
  ServerName www.sub1.domain.tld
  ServerPath /sub1/
  RewriteEngine On
  RewriteRule ^(/sub1/.* ) /www/subdomain$1
  # ...
</VirtualHost>

<VirtualHost 172.20.30.40>
  DocumentRoot /www/subdomain/sub2
  ServerName www.sub2.domain.tld
  ServerPath /sub2/
  RewriteEngine On
  RewriteRule ^(/sub2/.* ) /www/subdomain$1
  # ...
</VirtualHost>
```

ServerPath

URL http://www.sub1.domain.tld/s

subl- .

Host: ,

URL http://www.sub1.dc

subl- .

Host:

:

Host:

http://www.sub2.domain.tld/sub1/

subl- .

RewriteRule  
URL .

Host: (

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >



**1.3** .  
NameVirtualHost 1.3 .



<VirtualHost> . <VirtualHost>

Listen, ServerName, ServerPath, ServerAlias

. ( ) .

Listen 80. ServerPath ServerAlias  
ServerName IP .

Listen . .  
URI .

VirtualHost .  
\* . (DNS  
(address set) .

IP NameVirtualHost IP  
. IP \* .

IP NameVirtualHost  
NameVirtualHost (CNAME) IP NameVir

IP: NameVirtualHost ,  
NameVirtualHost VirtualHost .

NameVirtualHost VirtualHost  
( VirtualHost . ):

```
NameVirtualHost
111.22.33.44
<VirtualHost
111.22.33.44>
# A
```

```
<VirtualHost
111.22.33.44>
# A
</VirtualHost>
<VirtualHost
```

```

...
</VirtualHost>
<VirtualHost
111.22.33.44>
# B
...
</VirtualHost>

NameVirtualHost
111.22.33.55
<VirtualHost
111.22.33.55>
# C
...
</VirtualHost>
<VirtualHost
111.22.33.55>
# D
...
</VirtualHost>

```

```

111.22.33.55>
# C
...
</VirtualHost>
<VirtualHost
111.22.33.44>
# B
...
</VirtualHost>
<VirtualHost
111.22.33.55>
# D
...
</VirtualHost>

NameVirtualHost
111.22.33.44
NameVirtualHost
111.22.33.55

```

( .)

VirtualHost ,  
Listen .

VirtualHost

VirtualHost  
ServerAlias ).

ServerAlias  
Listen

.

IP .

NameVirtualHost I

.

. IP

IP

. IP

::

1. [ServerAdmin](#), [ResourceConfig](#), [AccessConfig](#),  
[Timeout](#), [KeepAliveTimeout](#), [KeepAlive](#),

MaxKeepAliveRequests, SendBufferSize

. (, .)

2. " (lookup defaults)" .  
(per-directory configuration) .

3. (per-server config) .

... ..  
.

ServerName .

IP .

ServerName

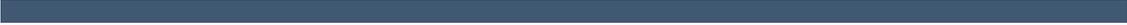
VirtualHost

.

\_default\_

ServerName





:

IP IP

IP  
\_default\_

IP

NameVirtualHost \* .

( IP ),

IP .

**IP**

IP .

, .

( IP  
Host:

) ,

Host: ,

ServerName ServerAlias

. Host:

Host: HTTP/1.0

ServerPath .

, ( )

IP

IP TCP/IP , KeepAlive/ ..

## URI

URI URI ,  
URI // URI .

- IP . IP NameVirtualHost
- IP ServerAlias ServerPath .
- , IP, \_default\_, Na
- Host:
- ( Host: ,) ServerPath  
ServerPath
- IP ,
- \_default\_ IP  
\_default\_ ( Listen)  
( ,\_default\_:\*)  
NameVirtualHost \* .
- IP ( \_default\_ )  
.. ( \_default\_ ) /
- ( ,NameVirtualHost ) ()  
Host: \_default\_

- DNS VirtualHost DNS .  
DNS .
- ServerName . DNS .







| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## (file descriptor)

- handle) ) .  
10-20 .  
hard-limit .
- , (file  
:  
1. setrlimit() .  
2. (Solaris 2.3) setrlimit(RLIMIT\_NOFILE)  
3. hard limit .  
4. (Solaris 2) stdio 256

:

- . <VirtualHost> . (
- () 12 ,

```
#!/bin/sh  
ulimit -S -n 100  
exec httpd
```







| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# DNS



.  
( )  
(theft of service)

. ( DNS .



```
<VirtualHost www.abc.dom>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

IP . IP , DNS .  
DNS www .  
( 1.2 . )  
www.abc.dom 192.0.2.1 . :

```
<VirtualHost 192.0.2.1>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

ServerName DNS .  
( 1.2 . ) , ,  
URL URL .  
.

```
<VirtualHost 192.0.2.1>
ServerName www.abc.dom
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```



( ... )  
0 . 1.2  
DNS . ,  
abc.dom DNS , www.abc.dom  
) .

```
<VirtualHost www.abc.dom>  
  ServerAdmin webgirl@abc.dom  
  DocumentRoot /www/abc  
</VirtualHost>  
  
<VirtualHost www.def.dom>  
  ServerAdmin webguy@def.dom  
  DocumentRoot /www/def  
</VirtualHost>
```

www.abc.dom 192.0.2.1, www.def.dom 192.0.2.2  
. , def.dom DNS . def.dor  
abc.dom . www.  
192.0.2.1 . DNS  
www.def.dom .

( http://www.abc.dom/whatever URL )  
192.0.2.1 def.dom .



```
1.1 ServerName C gethostname ( "hostname"
). DNS . .

DNS /etc/hosts .
.) DNS /etc
/etc/resolv.conf /etc/nsswitch.conf .

DNS HOSTRESORDER "local"
. mod_env CGI . manpage
FAQ .
```



- 
- VirtualHost IP

- Listen IP

- ServerName

- `<VirtualHost _default_*>`



DNS . 1.2  
. IP

DNS

IP DNS

DNS . (FTP  
)

TCP wrapper "-" DNS

IP DNS

HTTP/1.1

Host IP

DNS . 1997 3



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: An Introduction

As an introduction this chapter is aimed at readers who are familiar with the Web, HTTP, and Apache, but are not security experts. It is not intended to be a definitive guide to the SSL protocol, nor does it discuss specific techniques for managing certificates in an organization, or the important legal issues of patents and import and export restrictions. Rather, it is intended to provide a common background to `mod_ssl` users by pulling together various concepts, definitions, and examples as a starting point for further exploration.



Understanding SSL requires an understanding of cryptographic algorithms, message digest functions (aka. one-way or hash functions), and digital signatures. These techniques are the subject of entire books (see for instance [\[AC96\]](#)) and provide the basis for privacy, integrity, and authentication.

### **Cryptographic Algorithms**

Suppose Alice wants to send a message to her bank to transfer some money. Alice would like the message to be private, since it will include information such as her account number and transfer amount. One solution is to use a cryptographic algorithm, a technique that would transform her message into an encrypted form, unreadable until it is decrypted. Once in this form, the message can only be decrypted by using a secret key. Without the key the message is useless: good cryptographic algorithms make it so difficult for intruders to decode the original text that it isn't worth their effort.

There are two categories of cryptographic algorithms: conventional and public key.

#### **Conventional cryptography**

also known as symmetric cryptography, requires the sender and receiver to share a key: a secret piece of information that may be used to encrypt or decrypt a message. As long as this key is kept secret, nobody other than the sender or recipient can read the message. If Alice and the bank know a secret key, then they can send each other private messages. The task of sharing a key between sender and recipient before communicating, while also keeping it secret from others, can be problematic.

#### **Public key cryptography**

also known as asymmetric cryptography, solves the key

exchange problem by defining an algorithm which uses two keys, each of which may be used to encrypt a message. If one key is used to encrypt a message then the other must be used to decrypt it. This makes it possible to receive secure messages by simply publishing one key (the public key) and keeping the other secret (the private key).

Anyone can encrypt a message using the public key, but only the owner of the private key will be able to read it. In this way, Alice can send private messages to the owner of a key-pair (the bank), by encrypting them using their public key. Only the bank will be able to decrypt them.

## **Message Digests**

Although Alice may encrypt her message to make it private, there is still a concern that someone might modify her original message or substitute it with a different one, in order to transfer the money to themselves, for instance. One way of guaranteeing the integrity of Alice's message is for her to create a concise summary of her message and send this to the bank as well. Upon receipt of the message, the bank creates its own summary and compares it with the one Alice sent. If the summaries are the same then the message has been received intact.

A summary such as this is called a *message digest*, *one-way function* or *hash function*. Message digests are used to create a short, fixed-length representation of a longer, variable-length message. Digest algorithms are designed to produce a unique digest for each message. Message digests are designed to make it impractically difficult to determine the message from the digest and (in theory) impossible to find two different messages which create the same digest -- thus eliminating the possibility of substituting one message for another while maintaining the same digest.

Another challenge that Alice faces is finding a way to send the digest to the bank securely; if the digest is not sent securely, its integrity may be compromised and with it the possibility for the bank to determine the integrity of the original message. Only if the digest is sent securely can the integrity of the associated message be determined.

One way to send the digest securely is to include it in a digital signature.

## Digital Signatures

When Alice sends a message to the bank, the bank needs to ensure that the message is really from her, so an intruder cannot request a transaction involving her account. A *digital signature*, created by Alice and included with the message, serves this purpose.

Digital signatures are created by encrypting a digest of the message and other information (such as a sequence number) with the sender's private key. Though anyone can *decrypt* the signature using the public key, only the sender knows the private key. This means that only the sender can have signed the message. Including the digest in the signature means the signature is only good for that message; it also ensures the integrity of the message since no one can change the digest and still sign it.

To guard against interception and reuse of the signature by an intruder at a later date, the signature contains a unique sequence number. This protects the bank from a fraudulent claim from Alice that she did not send the message -- only she could have signed it (non-repudiation).



Although Alice could have sent a private message to the bank, signed it and ensured the integrity of the message, she still needs to be sure that she is really communicating with the bank. This means that she needs to be sure that the public key she is using is part of the bank's key-pair, and not an intruder's. Similarly, the bank needs to verify that the message signature really was signed by the private key that belongs to Alice.

If each party has a certificate which validates the other's identity, confirms the public key and is signed by a trusted agency, then both can be assured that they are communicating with whom they think they are. Such a trusted agency is called a *Certificate Authority* and certificates are used for authentication.

## Certificate Contents

A certificate associates a public key with the real identity of an individual, server, or other entity, known as the subject. As shown in [Table 1](#), information about the subject includes identifying information (the distinguished name) and the public key. It also includes the identification and signature of the Certificate Authority that issued the certificate and the period of time during which the certificate is valid. It may have additional information (or extensions) as well as administrative information for the Certificate Authority's use, such as a serial number.

**Table 1: Certificate Information**

<b>Subject</b>	Distinguished Name, Public Key
<b>Issuer</b>	Distinguished Name, Signature
<b>Period of Validity</b>	Not Before Date, Not After Date
<b>Administrative Information</b>	Version, Serial Number
<b>Extended Information</b>	Basic Constraints, Netscape Flags,

etc.

A distinguished name is used to provide an identity in a specific context -- for instance, an individual might have a personal certificate as well as one for their identity as an employee. Distinguished names are defined by the X.509 standard [X509], which defines the fields, field names and abbreviations used to refer to the fields (see [Table 2](#)).

**Table 2: Distinguished Name Information**

DN Field	Abbrev.	Description	Example
Common Name	CN	Name being certified	CN=Joe Average
Organization or Company	O	Name is associated with this organization	O=Snake Oil, Ltd.
Organizational Unit	OU	Name is associated with this organization unit, such as a department	OU=Research Institute
City/Locality	L	Name is located in this City	L=Snake City
State/Province	ST	Name is located in this State/Province	ST=Desert
Country	C	Name is located in this Country (ISO code)	C=XZ

A Certificate Authority may define a policy specifying which distinguished field names are optional and which are required. It may also place requirements upon the field contents, as may users of certificates. For example, a Netscape browser requires that the Common Name for a certificate representing a server matches a wildcard pattern for the domain name of that server,

such as \*.snakeoil.com.

The binary format of a certificate is defined using the ASN.1 notation [ASN1] [PKCS]. This notation defines how to specify the contents and encoding rules define how this information is translated into binary form. The binary encoding of the certificate is defined using Distinguished Encoding Rules (DER), which are based on the more general Basic Encoding Rules (BER). For those transmissions which cannot handle binary, the binary form may be translated into an ASCII form by using Base64 encoding [MIME]. When placed between begin and end delimiter lines (as below), this encoded version is called a PEM ("Privacy Enhanced Mail") encoded certificate.

### Example of a PEM-encoded certificate (snakeoil.crt)

```
-----BEGIN CERTIFICATE-----
MIIC7jCCALegAwIBAgIBATANBgkqhkiG9w0BAQQFADCbqTELMaKGA1UEBhMCWFkx
FTATBgNVBAGTDFNuYWt1IERlc2VydDETMBEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChMOU25ha2UgT21sLCBmdGQxHjAcBgNVBAsTFUN1cnRpZm1jYXR1IEF1dGhv
cm10eTEVMBMGA1UEAxMMU25ha2UgT21sIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTgxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAKGA1UEBhMCWFkxFTATBgNVBAGTDFNuYWt1IERlc2VydDETMBEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChMOU25ha2UgT21sLCBmdGQxHjAcBgNVBAsTD1d1YnN1
cnZlciBUZWFtMRkwFwYDVQQDExB3d3cuc25ha2VvaWwuZG9tMR8wHQYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zch+da+rPTx/DPRp3xGjHZ4GG6pCmvADIETBtKBFACZ64n+Dy7Np8b
vKR+yy5DGqiijsh1D/j8H1GE+q4TZ80Fk7BNBFazHxFbYI40KMicXdKzdif1yfaa
lWoANf1Az1SdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWGCSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAA0B
gQAZUIHAL4D09oE6Lv2k56Gp380BDuILvwLg1v1KL8mQR+KFjghCrtppqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdKpfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHziCxBVC1lnHyYgJDuAMhe3961YAn8bClD1/L4NMGBCQ==
-----END CERTIFICATE-----
```

## Certificate Authorities

By verifying the information in a certificate request before granting the certificate, the Certificate Authority assures itself of the identity of the private key owner of a key-pair. For instance, if Alice

requests a personal certificate, the Certificate Authority must first make sure that Alice really is the person the certificate request claims she is.

### **Certificate Chains**

A Certificate Authority may also issue a certificate for another Certificate Authority. When examining a certificate, Alice may need to examine the certificate of the issuer, for each parent Certificate Authority, until reaching one which she has confidence in. She may decide to trust only certificates with a limited chain of issuers, to reduce her risk of a "bad" certificate in the chain.

### **Creating a Root-Level CA**

As noted earlier, each certificate requires an issuer to assert the validity of the identity of the certificate subject, up to the top-level Certificate Authority (CA). This presents a problem: who can vouch for the certificate of the top-level authority, which has no issuer? In this unique case, the certificate is "self-signed", so the issuer of the certificate is the same as the subject. Browsers are preconfigured to trust well-known certificate authorities, but it is important to exercise extra care in trusting a self-signed certificate. The wide publication of a public key by the root authority reduces the risk in trusting this key -- it would be obvious if someone else publicized a key claiming to be the authority.

A number of companies, such as [Thawte](#) and [VeriSign](#) have established themselves as Certificate Authorities. These companies provide the following services:

- Verifying certificate requests
- Processing certificate requests
- Issuing and managing certificates

It is also possible to create your own Certificate Authority. Although risky in the Internet environment, it may be useful within

an Intranet where the organization can easily verify the identities of individuals and servers.

### **Certificate Management**

Establishing a Certificate Authority is a responsibility which requires a solid administrative, technical and management framework. Certificate Authorities not only issue certificates, they also manage them -- that is, they determine for how long certificates remain valid, they renew them and keep lists of certificates that were issued in the past but are no longer valid (Certificate Revocation Lists, or CRLs).

For example, if Alice is entitled to a certificate as an employee of a company but has now left that company, her certificate may need to be revoked. Because certificates are only issued after the subject's identity has been verified and can then be passed around to all those with whom the subject may communicate, it is impossible to tell from the certificate alone that it has been revoked. Therefore when examining certificates for validity it is necessary to contact the issuing Certificate Authority to check CRLs -- this is usually not an automated part of the process.

#### **Note**

If you use a Certificate Authority that browsers are not configured to trust by default, it is necessary to load the Certificate Authority certificate into the browser, enabling the browser to validate server certificates signed by that Certificate Authority. Doing so may be dangerous, since once loaded, the browser will accept all certificates signed by that Certificate Authority.



The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP). SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity and encryption for privacy.

The protocol is designed to support a range of choices for specific algorithms used for cryptography, digests and signatures. This allows algorithm selection for specific servers to be made based on legal, export or other concerns and also enables the protocol to take advantage of new algorithms. Choices are negotiated between client and server when establishing a protocol session.

**Table 4: Versions of the SSL protocol**

Version	Source	Description
SSL v2.0	Vendor Standard (from Netscape Corp.)	First SSL protocol for which implementations exist
SSL v3.0	Expired Internet Draft (from Netscape Corp.) [ <a href="#">SSL3</a> ]	Revisions to prevent specific security attacks, add non-RSA ciphers and support for certificate chains
TLS v1.0	Proposed Internet Standard (from IETF) [ <a href="#">TLS1</a> ]	Revision of SSL 3.0 to update the MAC layer to HMAC, add block padding for block ciphers, message order standardization and more alert messages.
TLS	Proposed	Update of TLS 1.0 to add protection

v1.1	Internet Standard (from IETF) <a href="#">[TLS11]</a>	against Cipher block chaining (CBC) attacks.
TLS v1.2	Proposed Internet Standard (from IETF) <a href="#">[TLS12]</a>	Update of TLS 1.1 deprecating MD5 as hash, and adding incompatibility to SSL so it will never negotiate the use of SSLv2.

There are a number of versions of the SSL protocol, as shown in [Table 4](#). As noted there, one of the benefits in SSL 3.0 is that it adds support of certificate chain loading. This feature allows a server to pass a server certificate along with issuer certificates to the browser. Chain loading also permits the browser to validate the server certificate, even if Certificate Authority certificates are not installed for the intermediate issuers, since they are included in the certificate chain. SSL 3.0 is the basis for the Transport Layer Security [\[TLS\]](#) protocol standard, currently in development by the Internet Engineering Task Force (IETF).

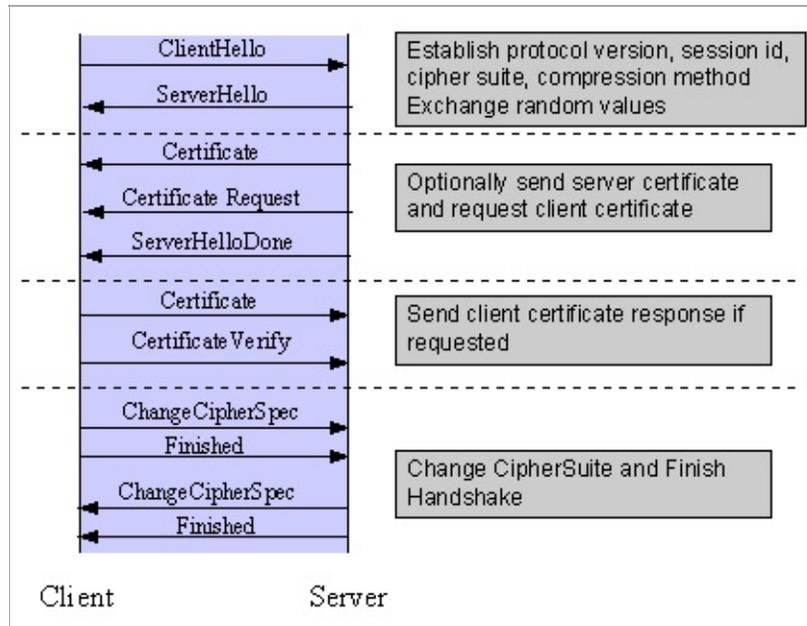
## Establishing a Session

The SSL session is established by following a handshake sequence between client and server, as shown in [Figure 1](#). This sequence may vary, depending on whether the server is configured to provide a server certificate or request a client certificate. Although cases exist where additional handshake steps are required for management of cipher information, this article summarizes one common scenario. See the SSL specification for the full range of possibilities.

### Note

Once an SSL session has been established, it may be reused.

This avoids the performance penalty of repeating the many steps needed to start a session. To do this, the server assigns each SSL session a unique session identifier which is cached in the server and which the client can use in future connections to reduce the handshake time (until the session identifier expires from the cache of the server).



**Figure 1:** Simplified SSL Handshake Sequence

The elements of the handshake sequence, as used by the client and server, are listed below:

1. Negotiate the Cipher Suite to be used during data transfer
2. Establish and share a session key between client and server
3. Optionally authenticate the server to the client
4. Optionally authenticate the client to the server

The first step, Cipher Suite Negotiation, allows the client and server to choose a Cipher Suite supported by both of them. The SSL3.0 protocol specification defines 31 Cipher Suites. A Cipher Suite is defined by the following components:

- Key Exchange Method
- Cipher for Data Transfer
- Message Digest for creating the Message Authentication Code (MAC)

These three elements are described in the sections that follow.

## Key Exchange Method

The key exchange method defines how the shared secret symmetric cryptography key used for application data transfer will be agreed upon by client and server. SSL 2.0 uses RSA key exchange only, while SSL 3.0 supports a choice of key exchange algorithms including RSA key exchange (when certificates are used), and Diffie-Hellman key exchange (for exchanging keys without certificates, or without prior communication between client and server).

One variable in the choice of key exchange methods is digital signatures -- whether or not to use them, and if so, what kind of signatures to use. Signing with a private key provides protection against a man-in-the-middle-attack during the information exchange used to generating the shared key [[AC96](#), p516].

## Cipher for Data Transfer

SSL uses conventional symmetric cryptography, as described earlier, for encrypting messages in a session. There are nine choices of how to encrypt, including the option not to encrypt:

- No encryption
- Stream Ciphers
  - RC4 with 40-bit keys
  - RC4 with 128-bit keys
- CBC Block Ciphers

- RC2 with 40 bit key
- DES with 40 bit key
- DES with 56 bit key
- Triple-DES with 168 bit key
- Idea (128 bit key)
- Fortezza (96 bit key)

"CBC" refers to Cipher Block Chaining, which means that a portion of the previously encrypted cipher text is used in the encryption of the current block. "DES" refers to the Data Encryption Standard [[AC96](#), ch12], which has a number of variants (including DES40 and 3DES\_EDE). "Idea" is currently one of the best and cryptographically strongest algorithms available, and "RC2" is a proprietary algorithm from RSA DSI [[AC96](#), ch13].

## Digest Function

The choice of digest function determines how a digest is created from a record unit. SSL supports the following:

- No digest (Null choice)
- MD5, a 128-bit hash
- Secure Hash Algorithm (SHA-1), a 160-bit hash

The message digest is used to create a Message Authentication Code (MAC) which is encrypted with the message to verify integrity and to protect against replay attacks.

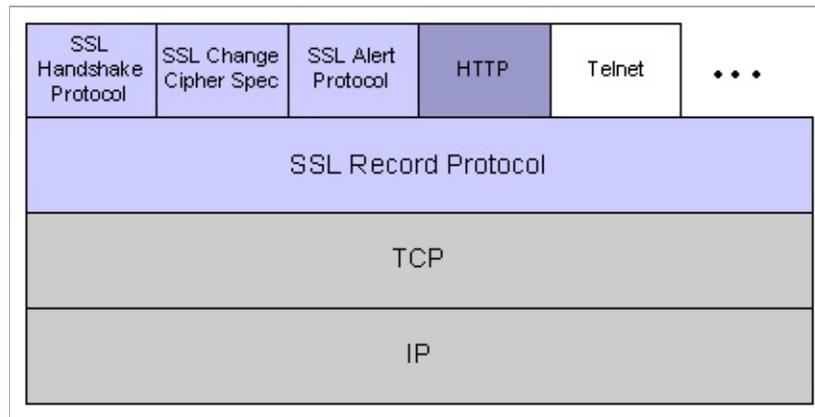
## Handshake Sequence Protocol

The handshake sequence uses three protocols:

- The *SSL Handshake Protocol* for performing the client and server SSL session establishment.
- The *SSL Change Cipher Spec Protocol* for actually establishing agreement on the Cipher Suite for the session.

- The *SSL Alert Protocol* for conveying SSL error messages between client and server.

These protocols, as well as application protocol data, are encapsulated in the *SSL Record Protocol*, as shown in [Figure 2](#). An encapsulated protocol is transferred as data by the lower layer protocol, which does not examine the data. The encapsulated protocol has no knowledge of the underlying protocol.



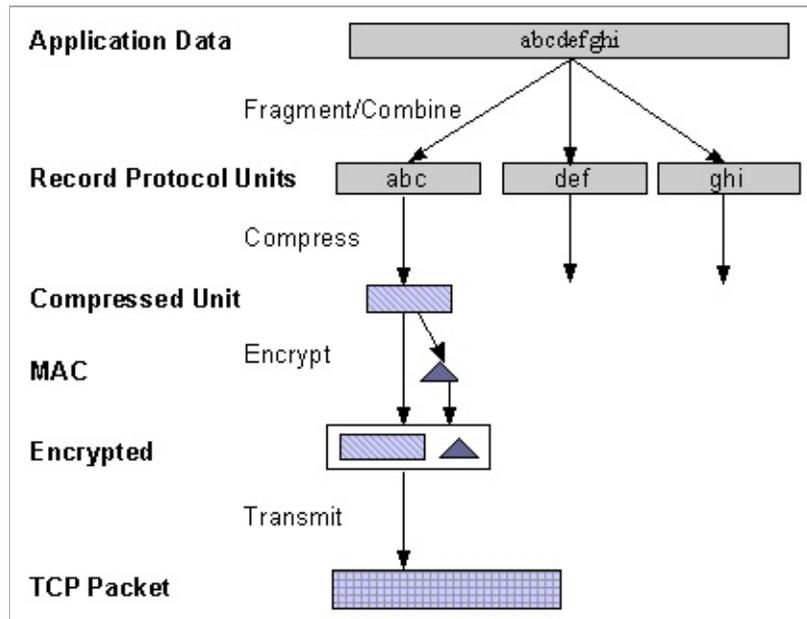
**Figure 2:** *SSL Protocol Stack*

The encapsulation of SSL control protocols by the record protocol means that if an active session is renegotiated the control protocols will be transmitted securely. If there was no previous session, the Null cipher suite is used, which means there will be no encryption and messages will have no integrity digests, until the session has been established.

## Data Transfer

The SSL Record Protocol, shown in [Figure 3](#), is used to transfer application and SSL Control data between the client and server, where necessary fragmenting this data into smaller units, or combining multiple higher level protocol data messages into single units. It may compress, attach digest signatures, and encrypt these units before transmitting them using the underlying reliable transport protocol (Note: currently, no major SSL implementations

include support for compression).



**Figure 3: SSL Record Protocol**

## Securing HTTP Communication

One common use of SSL is to secure Web HTTP communication between a browser and a webserver. This does not preclude the use of non-secured HTTP - the secure version (called HTTPS) is the same as plain HTTP over SSL, but uses the URL scheme `https` rather than `http`, and a different server port (by default, port 443). This functionality is a large part of what [mod\\_ssl](#) provides for the Apache webserver.



## [AC96]

Bruce Schneier, *“Applied Cryptography”*, 2nd Edition, Wiley, 1996. See <http://www.counterpane.com/> for various other materials by Bruce Schneier.

## [ASN1]

ITU-T Recommendation X.208, *“Specification of Abstract Syntax Notation One (ASN.1)”*, last updated 2008. See <http://www.itu.int/ITU-T/asn1/>.

## [X509]

ITU-T Recommendation X.509, *“The Directory - Authentication Framework”*. For references, see <http://en.wikipedia.org/wiki/X.509>.

## [PKCS]

*“Public Key Cryptography Standards (PKCS)”*, RSA Laboratories Technical Notes, See <http://www.rsasecurity.com/rsalabs/pkcs/>.

## [MIME]

N. Freed, N. Borenstein, *“Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”*, RFC2045. See for instance <http://tools.ietf.org/html/rfc2045>.

## [SSL3]

Alan O. Freier, Philip Karlton, Paul C. Kocher, *“The SSL Protocol Version 3.0”*, 1996. See <http://www.netscape.com/eng/ssl3/draft302.txt>.

## [TLS1]

Tim Dierks, Christopher Allen, *“The TLS Protocol Version 1.0”*, 1999. See <http://ietf.org/rfc/rfc2246.txt>.

## [TLS11]

*“The TLS Protocol Version 1.1”*, 2006. See

<http://tools.ietf.org/html/rfc4346>.

**[TLS12]**

“*The TLS Protocol Version 1.2*”, 2008. See  
<http://tools.ietf.org/html/rfc5246>.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: Compatibility

This page covers backwards compatibility between `mod_ssl` and other SSL solutions. `mod_ssl` is not the only SSL solution for Apache; four additional products are (or were) also available: Ben Laurie's freely available [Apache-SSL](#) (from where `mod_ssl` were originally derived in 1998), Red Hat's commercial Secure Web Server (which was based on `mod_ssl`), Covalent's commercial Raven SSL Module (also based on `mod_ssl`) and finally C2Net's (now Red Hat's) commercial product [Stronghold](#) (based on a different evolution branch, named Sioux up to Stronghold 2.x, and based on `mod_ssl` since Stronghold 3.x).

`mod_ssl` mostly provides a superset of the functionality of all the other solutions, so it's simple to migrate from one of the older modules to `mod_ssl`. The configuration directives and environment variable names used by the older SSL solutions vary from those used in `mod_ssl`; mapping tables are included here to give the equivalents used by `mod_ssl`.



The mapping between configuration directives used by Apache-SSL 1.x and mod\_ssl 2.0.x is given in [Table 1](#). The mapping from Sioux 1.x and Stronghold 2.x is only partial because of special functionality in these interfaces which mod\_ssl doesn't provide.

**Table 1: Configuration Directive Mapping**

Old Directive	mod_ssl Directive
<b>Apache-SSL 1.x &amp; mod_ssl 2.0.x compatibility:</b>	
SSLEnable	SSL Engine on
SSLDisable	SSL Engine off
SSLLogFile <i>file</i>	
SSLRequiredCiphers <i>spec</i>	SSLCipherSuite <i>spec</i>
SSLRequireCipher <i>c1 ...</i>	SSLRequire %{SSL_CIPH {"c1", ...}}
SSLBanCipher <i>c1 ...</i>	SSLRequire not (%{SSL. in {"c1", ...})
SSLFakeBasicAuth	SSLOptions +FakeBasic
SSLCacheServerPath <i>dir</i>	-
SSLCacheServerPort <i>integer</i>	-
<b>Apache-SSL 1.x compatibility:</b>	
SSLExportClientCertificates	SSLOptions +ExportCer
SSLCacheServerRunDir <i>dir</i>	-
<b>Sioux 1.x compatibility:</b>	
SSL_CertFile <i>file</i>	SSLCertificateFile <i>file</i>
SSL_KeyFile <i>file</i>	SSLCertificateKeyFile
SSL_CipherSuite <i>arg</i>	SSLCipherSuite <i>arg</i>

SSL_X509VerifyDir <i>arg</i>	SSLCACertificatePath <i>a</i>
SSL_Log <i>file</i>	-
SSL_Connect <i>flag</i>	SSLEngine <i>flag</i>
SSL_ClientAuth <i>arg</i>	SSLVerifyClient <i>arg</i>
SSL_X509VerifyDepth <i>arg</i>	SSLVerifyDepth <i>arg</i>
SSL_FetchKeyPhraseFrom <i>arg</i>	-
SSL_SessionDir <i>dir</i>	-
SSL_Require <i>expr</i>	-
SSL_CertFileType <i>arg</i>	-
SSL_KeyFileType <i>arg</i>	-
SSL_X509VerifyPolicy <i>arg</i>	-
SSL_LogX509Attributes <i>arg</i>	-
<b>Stronghold 2.x compatibility:</b>	
StrongholdAccelerator <i>engine</i>	SSLCryptoDevice <i>engine</i>
StrongholdKey <i>dir</i>	-
StrongholdLicenseFile <i>dir</i>	-
SSLFlag <i>flag</i>	SSLEngine <i>flag</i>
SSLSessionLockFile <i>file</i>	SSLMutex <i>file</i>

SSLCipherList <i>spec</i>	SSLCipherSuite <i>spec</i>
RequireSSL	SSLRequireSSL
SSLLogFile <i>file</i>	-
SSLRoot <i>dir</i>	-
SSL_CertificateLogDir <i>dir</i>	-
AuthCertDir <i>dir</i>	-
SSL_Group <i>name</i>	-
SSLProxyMachineCertPath <i>dir</i>	SSLProxyMachineCertif <i>dir</i>
SSLProxyMachineCertFile <i>file</i>	SSLProxyMachineCertif <i>file</i>
SSLProxyCipherList <i>spec</i>	SSLProxyCipherSpec <i>spe</i>



The mapping between environment variable names used by the older SSL solutions and the names used by mod\_ssl is given in [Table 2](#).

**Table 2: Environment Variable Derivation**

Old Variable	mod_ssl Variable
SSL_PROTOCOL_VERSION	SSL_PROTOCOL
SSLEAY_VERSION	SSL_VERSION_LIBRAR
HTTPS_SECRETKEYSIZE	SSL_CIPHER_USEKEYS
HTTPS_KEYSIZE	SSL_CIPHER_ALGKEYS
HTTPS_CIPHER	SSL_CIPHER
HTTPS_EXPORT	SSL_CIPHER_EXPORT
SSL_SERVER_KEY_SIZE	SSL_CIPHER_ALGKEYS
SSL_SERVER_CERTIFICATE	SSL_SERVER_CERT
SSL_SERVER_CERT_START	SSL_SERVER_V_START
SSL_SERVER_CERT_END	SSL_SERVER_V_END
SSL_SERVER_CERT_SERIAL	SSL_SERVER_M_SERIA
SSL_SERVER_SIGNATURE_ALGORITHM	SSL_SERVER_A_SIG
SSL_SERVER_DN	SSL_SERVER_S_DN
SSL_SERVER_CN	SSL_SERVER_S_DN_CN
SSL_SERVER_EMAIL	SSL_SERVER_S_DN_Em
SSL_SERVER_O	SSL_SERVER_S_DN_O
SSL_SERVER_OU	SSL_SERVER_S_DN_OU
SSL_SERVER_C	SSL_SERVER_S_DN_C
SSL_SERVER_SP	SSL_SERVER_S_DN_SP
SSL_SERVER_L	SSL_SERVER_S_DN_L
SSL_SERVER_IDN	SSL_SERVER_I_DN
SSL_SERVER_ICN	SSL_SERVER_I_DN_CN
SSL_SERVER_IEMAIL	SSL_SERVER_I_DN_Em

SSL_SERVER_IO	SSL_SERVER_I_DN_O
SSL_SERVER_IOU	SSL_SERVER_I_DN_OU
SSL_SERVER_IC	SSL_SERVER_I_DN_C
SSL_SERVER_ISP	SSL_SERVER_I_DN_SP
SSL_SERVER_IL	SSL_SERVER_I_DN_L
SSL_CLIENT_CERTIFICATE	SSL_CLIENT_CERT
SSL_CLIENT_CERT_START	SSL_CLIENT_V_START
SSL_CLIENT_CERT_END	SSL_CLIENT_V_END
SSL_CLIENT_CERT_SERIAL	SSL_CLIENT_M_SERIA
SSL_CLIENT_SIGNATURE_ALGORITHM	SSL_CLIENT_A_SIG
SSL_CLIENT_DN	SSL_CLIENT_S_DN
SSL_CLIENT_CN	SSL_CLIENT_S_DN_CN
SSL_CLIENT_EMAIL	SSL_CLIENT_S_DN_Em
SSL_CLIENT_O	SSL_CLIENT_S_DN_O
SSL_CLIENT_OU	SSL_CLIENT_S_DN_OU
SSL_CLIENT_C	SSL_CLIENT_S_DN_C
SSL_CLIENT_SP	SSL_CLIENT_S_DN_SP
SSL_CLIENT_L	SSL_CLIENT_S_DN_L
SSL_CLIENT_IDN	SSL_CLIENT_I_DN
SSL_CLIENT_ICN	SSL_CLIENT_I_DN_CN
SSL_CLIENT_IEMAIL	SSL_CLIENT_I_DN_Em
SSL_CLIENT_IO	SSL_CLIENT_I_DN_O
SSL_CLIENT_IOU	SSL_CLIENT_I_DN_OU
SSL_CLIENT_IC	SSL_CLIENT_I_DN_C
SSL_CLIENT_ISP	SSL_CLIENT_I_DN_SP
SSL_CLIENT_IL	SSL_CLIENT_I_DN_L
SSL_EXPORT	SSL_CIPHER_EXPORT
SSL_KEYSIZE	SSL_CIPHER_ALGKEYS
SSL_SECKEYSIZE	SSL_CIPHER_USEKEYS
SSL_SSLEAY_VERSION	SSL_VERSION_LIBRAR

SSL_STRONG_CRYPTO	-
SSL_SERVER_KEY_EXP	-
SSL_SERVER_KEY_ALGORITHM	-
SSL_SERVER_KEY_SIZE	-
SSL_SERVER_SESSIONDIR	-
SSL_SERVER_CERTIFICATELOGDIR	-
SSL_SERVER_CERTFILE	-
SSL_SERVER_KEYFILE	-
SSL_SERVER_KEYFILETYPE	-

SSL_CLIENT_KEY_EXP	-
--------------------	---

SSL_CLIENT_KEY_ALGORITHM	-
--------------------------	---

SSL_CLIENT_KEY_SIZE	-
---------------------	---



## Custom Log Functions

When `mod_ssl` is enabled, additional functions exist for the [Custom Log Format](#) of `mod_log_config` as documented in the Reference Chapter. Beside the ``%{varname}x"` eXtension format function which can be used to expand any variables provided by any module, an additional Cryptography ``%{name}c"` cryptography format function exists for backward compatibility. The currently implemented function calls are listed in [Table 3](#).

**Table 3: Custom Log Cryptography Function**

Function Call	Description
<code>%...{version}c</code>	SSL protocol version
<code>%...{cipher}c</code>	SSL cipher
<code>%... {subjectdn}c</code>	Client Certificate Subject Distinguished Name
<code>%...{issuerdn}c</code>	Client Certificate Issuer Distinguished Name
<code>%...{errcode}c</code>	Certificate Verification Error (numerical)
<code>%...{errstr}c</code>	Certificate Verification Error (string)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: How-To

This document is intended to get you started, and get a few things working. You are strongly encouraged to read the rest of the SSL documentation, and arrive at a deeper understanding of the material, before progressing to the advanced techniques.



## Basic Configuration Example

Your SSL configuration will need to contain, at minimum, the following directives.

```
LoadModule ssl_module modules/mod_ssl.so

Listen 443
<VirtualHost *:443>
    ServerName www.example.com
    SSLEngine on
    SSLCertificateFile "/path/to/www.example.com/cert.pem"
    SSLCertificateKeyFile "/path/to/www.example.com/key.pem"
</VirtualHost>
```



- [How can I create an SSL server which accepts strong encryption only?](#)
- [How can I create an SSL server which accepts all types of ciphers in general, but requires a strong cipher for access to a particular URL?](#)

## How can I create an SSL server which accepts strong encryption only?

The following enables only the strongest ciphers:

```
SSLCipherSuite HIGH:!aNULL:!MD5
```

While with the following configuration you specify a preference for specific speed-optimized ciphers (which will be selected by `mod_ssl`, provided that they are supported by the client):

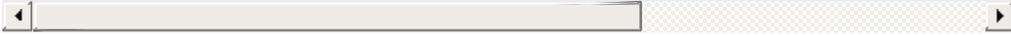
```
SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:!aNULL  
SSLHonorCipherOrder on
```

## How can I create an SSL server which accepts all types of ciphers in general, but requires a strong ciphers for access to a particular URL?

Obviously, a server-wide `SSLCipherSuite` which restricts ciphers to the strong variants, isn't the answer here. However, `mod_ssl` can be reconfigured within `Location` blocks, to give a per-directory solution, and can automatically force a renegotiation of the SSL parameters to meet the new configuration. This can be done as follows:

```
# be liberal in general
```

```
SSLCipherSuite ALL:!aNULL:RC4+RSA:+HIGH:+MEI  
  
<Location "/strong/area">  
# but https://hostname/strong/area/ and below  
# requires strong ciphers  
SSLCipherSuite HIGH:!aNULL:!MD5  
</Location>
```



## OCSP Stapling

The Online Certificate Status Protocol (OCSP) is a mechanism for determining whether or not a server certificate has been revoked, and OCSP Stapling is a special form of this in which the server, such as `httpd` and `mod_ssl`, maintains current OCSP responses for its certificates and sends them to clients which communicate with the server. Most certificates contain the address of an OCSP responder maintained by the issuing Certificate Authority, and `mod_ssl` can communicate with that responder to obtain a signed response that can be sent to clients communicating with the server.

Because the client can obtain the certificate revocation status from the server, without requiring an extra connection from the client to the Certificate Authority, OCSP Stapling is the preferred way for the revocation status to be obtained. Other benefits of eliminating the communication between clients and the Certificate Authority are that the client browsing history is not exposed to the Certificate Authority and obtaining status is more reliable by not depending on potentially heavily loaded Certificate Authority servers.

Because the response obtained by the server can be reused for all clients using the same certificate during the time that the response is valid, the overhead for the server is minimal.

Once general SSL support has been configured properly, enabling OCSP Stapling generally requires only very minor modifications to the `httpd` configuration — the addition of these two directives:

```
SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
```

These directives are placed at global scope (i.e., not within a virtual host definition) wherever other global SSL configuration

directives are placed, such as in `conf/extra/httpd-ssl.conf` for normal open source builds of `httpd`, `/etc/apache2/mods-enabled/ssl.conf` for the Ubuntu or Debian-bundled `httpd`, etc.

The path on the `SSLStaplingCache` directive (e.g., `logs/`) should match the one on the `SSLSessionCache` directive. This path is relative to `ServerRoot`.

This particular `SSLStaplingCache` directive requires `mod_socache_shmcb` (from the `shmcb` prefix on the directive's argument). This module is usually enabled already for `SSLSessionCache` or on behalf of some module other than `mod_ssl`. If you enabled an SSL session cache using a mechanism other than `mod_socache_shmcb`, use that alternative mechanism for `SSLStaplingCache` as well. For example:

```
SSLSessionCache "dbm:logs/ssl_scache"  
SSLStaplingCache "dbm:logs/ssl_stapling"
```

You can use the `openssl` command-line program to verify that an OCSP response is sent by your server:

```
$ openssl s_client -connect www.example.com:443 -...  
...  
OCSP response:  
=====  
OCSP Response Data:  
    OCSP Response Status: successful (0x0)  
    Response Type: Basic OCSP Response  
...  
    Cert Status: Good  
...
```

The following sections highlight the most common situations which require further modification to the configuration. Refer also to the

[mod\\_ssl](#) reference manual.

## If more than a few SSL certificates are used for the server

OCSP responses are stored in the SSL stapling cache. While the responses are typically a few hundred to a few thousand bytes in size, `mod_ssl` supports OCSP responses up to around 10K bytes in size. With more than a few certificates, the stapling cache size (32768 bytes in the example above) may need to be increased. Error message AH01929 will be logged in case of an error storing a response.

## If the certificate does not point to an OCSP responder, or if a different address must be used

Refer to the [SSLStaplingForceURL](#) directive.

You can confirm that a server certificate points to an OCSP responder using the `openssl` command-line program, as follows:

```
$ openssl x509 -in ./www.example.com.crt -text | \
OCSP - URI:http://ocsp.example.com
```

If the OCSP URI is provided and the web server can communicate to it directly without using a proxy, no configuration is required. Note that firewall rules that control outbound connections from the web server may need to be adjusted.

If no OCSP URI is provided, contact your Certificate Authority to determine if one is available; if so, configure it with [SSLStaplingForceURL](#) in the virtual host that uses the certificate.

## If multiple SSL-enabled virtual hosts are configured and OCSP Stapling should be disabled for some

Add `SSLUseStapling Off` to the virtual hosts for which OCSP Stapling should be disabled.

### **If the OCSP responder is slow or unreliable**

Several directives are available to handle timeouts and errors. Refer to the documentation for the [SSLStaplingFakeTryLater](#), [SSLStaplingResponderTimeout](#), and [SSLStaplingReturnResponderErrors](#) directives.

### **If mod\_ssl logs error AH02217**

AH02217: ssl\_stapling\_init\_cert: Can't retrieve i

In order to support OCSP Stapling when a particular server certificate is used, the certificate chain for that certificate must be configured. If it was not configured as part of enabling SSL, the AH02217 error will be issued when stapling is enabled, and an OCSP response will not be provided for clients using the certificate.

Refer to the [SSLCertificateChainFile](#) and [SSLCertificateFile](#) for instructions for configuring the certificate chain.



- [How can I force clients to authenticate using certificates?](#)
- [How can I force clients to authenticate using certificates for a particular URL, but still allow arbitrary clients to access the rest of the server?](#)
- [How can I allow only clients who have certificates to access a particular URL, but allow all clients to access the rest of the server?](#)
- [How can I require HTTPS with strong ciphers, and either basic authentication or client certificates, for access to part of the Intranet website, for clients coming from the Internet?](#)

## How can I force clients to authenticate using certificates?

When you know all of your users (eg, as is often the case on a corporate Intranet), you can require plain certificate authentication. All you need to do is to create client certificates signed by your own CA certificate (ca.crt) and then verify the clients against this certificate.

```
# require a client certificate which has to  
# signed by our CA certificate in ca.crt  
SSLVerifyClient require  
SSLVerifyDepth 1  
SSLCACertificateFile "conf/ssl.crt/ca.crt"
```

## How can I force clients to authenticate using certificates for a particular URL, but still allow arbitrary clients to access the rest of the server?

To force clients to authenticate using certificates for a particular URL, you can use the per-directory reconfiguration features of [mod\\_ssl](#):

```
SSLVerifyClient none
SSLCACertificateFile "conf/ssl.crt/ca.crt"

<Location "/secure/area">
SSLVerifyClient require
SSLVerifyDepth 1
</Location>
```

## How can I allow only clients who have certificates to access a particular URL, but allow all clients to access the rest of the server?

The key to doing this is checking that part of the client certificate matches what you expect. Usually this means checking all or part of the Distinguished Name (DN), to see if it contains some known string. There are two ways to do this, using either [mod\\_auth\\_basic](#) or [SSLRequire](#).

The [mod\\_auth\\_basic](#) method is generally required when the certificates are completely arbitrary, or when their DNs have no common fields (usually the organisation, etc.). In this case, you should establish a password database containing *all* clients allowed, as follows:

```
SSLVerifyClient      none
SSLCACertificateFile "conf/ssl.crt/ca.crt"
SSLCACertificatePath "conf/ssl.crt"

<Directory "/usr/local/apache2/htdocs/secure"
    SSLVerifyClient      require
    SSLVerifyDepth      5
    SSLOptions           +FakeBasicAuth
    SSLRequireSSL
    AuthName             "Snake Oil Authent:
    AuthType             Basic
```

```
AuthBasicProvider    file
AuthUserFile         "/usr/local/apache2/
Require              valid-user
</Directory>
```

The password used in this example is the DES encrypted string "password". See the [SSLOptions](#) docs for more information.

### httpd.passwd

```
/C=DE/L=Munich/O=Snake Oil, Ltd./OU=Staff/CN=Foo:xxj31ZMTZzkVA
/C=US/L=S.F./O=Snake Oil, Ltd./OU=CA/CN=Bar:xxj31ZMTZzkVA
/C=US/L=L.A./O=Snake Oil, Ltd./OU=Dev/CN=Quux:xxj31ZMTZzkVA
```

When your clients are all part of a common hierarchy, which is encoded into the DN, you can match them more easily using [SSLRequire](#), as follows:

```
SSLVerifyClient      none
SSLCACertificateFile "conf/ssl.crt/ca.crt"
SSLCACertificatePath "conf/ssl.crt"

<Directory "/usr/local/apache2/htdocs/secure">
  SSLVerifyClient      require
  SSLVerifyDepth       5
  SSLOptions           +FakeBasicAuth
  SSLRequireSSL
  SSLRequire           %{SSL_CLIENT_S_DN_O} eq
                       and %{SSL_CLIENT_S_DN_OU} in
</Directory>
```

**How can I require HTTPS with strong ciphers, and either basic authentication or client certificates, for access to part of the Intranet website, for clients**

## coming from the Internet? I still want to allow plain HTTP access for clients on the Intranet.

These examples presume that clients on the Intranet have IPs in the range 192.168.1.0/24, and that the part of the Intranet website you want to allow internet access to is /usr/local/apache2/htdocs/subarea. This configuration should remain outside of your HTTPS virtual host, so that it applies to both HTTPS and HTTP.

```
SSLCACertificateFile "conf/ssl.crt/company-c

<Directory "/usr/local/apache2/htdocs">
    #   Outside the subarea only Intranet ac
    Require          ip 192.168.1.0/24
</Directory>

<Directory "/usr/local/apache2/htdocs/subare
    #   Inside the subarea any Intranet acc
    #   but from the Internet only HTTPS + S
    #   or the alternative HTTPS + Strong-C

    #   If HTTPS is used, make sure a strong
    #   Additionally allow client certs as a
    SSLVerifyClient  optional
    SSLVerifyDepth  1
    SSLOptions       +FakeBasicAuth +St
    SSLRequire       %{SSL_CIPHER_USEKEY

    #   Force clients from the Internet to u
    RewriteEngine    on
    RewriteCond      "%{REMOTE_ADDR}" "
    RewriteCond      "%{HTTPS}" " !=on"
    RewriteRule      "." "-" [F]

    #   Allow Network Access and/or Basic Au
    Satisfy          any
```

```
# Network Access Control
Require ip 192.168.1.0/24

# HTTP Basic Authentication
AuthType basic
AuthName "Protected Intranet"
AuthBasicProvider file
AuthUserFile "conf/protected.passwd"
Require valid-user
</Directory>
```



`mod_ssl` can log extremely verbose debugging information to the error log, when its `LogLevel` is set to the higher trace levels. On the other hand, on a very busy server, level `info` may already be too much. Remember that you can configure the `LogLevel` per module to suite your needs.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: FAQ

*The wise man doesn't give the right answers, he poses the right questions.*

-- Claude Levi-Strauss



- [Why do I get permission errors related to SSLMutex when I start Apache?](#)
- [Why does mod\\_ssl stop with the error "Failed to generate temporary 512 bit RSA private key" when I start Apache?](#)

## Why do I get permission errors related to SSLMutex when I start Apache?

Errors such as ``mod_ssl: Child could not open SSLMutex lockfile /opt/apache/logs/ssl_mutex.18332 (System error follows) [...] System: Permission denied (errno: 13)"` are usually caused by overly restrictive permissions on the *parent* directories. Make sure that all parent directories (here `/opt`, `/opt/apache` and `/opt/apache/logs`) have the x-bit set for, at minimum, the UID under which Apache's children are running (see the [User](#) directive).

## Why does mod\_ssl stop with the error "Failed to generate temporary 512 bit RSA private key" when I start Apache?

Cryptographic software needs a source of unpredictable data to work correctly. Many open source operating systems provide a "randomness device" that serves this purpose (usually named `/dev/random`). On other systems, applications have to seed the OpenSSL Pseudo Random Number Generator (PRNG) manually with appropriate data before generating keys or performing public key encryption. As of version 0.9.5, the OpenSSL functions that need randomness report an error if the PRNG has not been seeded with at least 128 bits of randomness.

To prevent this error, `mod_ssl` has to provide enough entropy to the PRNG to allow it to work correctly. This can be done via the

SSLRandomSeed directive.



- [Is it possible to provide HTTP and HTTPS from the same server?](#)
- [Which port does HTTPS use?](#)
- [How do I speak HTTPS manually for testing purposes?](#)
- [Why does the connection hang when I connect to my SSL-aware Apache server?](#)
- [Why do I get "Connection Refused" errors, when trying to access my newly installed Apache+mod\\_ssl server via HTTPS?](#)
- [Why are the SSL\\_XXX variables not available to my CGI & SSI scripts?](#)
- [How can I switch between HTTP and HTTPS in relative hyperlinks?](#)

## **Is it possible to provide HTTP and HTTPS from the same server?**

Yes. HTTP and HTTPS use different server ports (HTTP binds to port 80, HTTPS to port 443), so there is no direct conflict between them. You can either run two separate server instances bound to these ports, or use Apache's elegant virtual hosting facility to create two virtual servers, both served by the same instance of Apache - one responding over HTTP to requests on port 80, and the other responding over HTTPS to requests on port 443.

## **Which port does HTTPS use?**

You can run HTTPS on any port, but the standards specify port 443, which is where any HTTPS compliant browser will look by default. You can force your browser to look on a different port by specifying it in the URL. For example, if your server is set up to serve pages over HTTPS on port 8080, you can access them at `https://example.com:8080/`

## How do I speak HTTPS manually for testing purposes?

While you usually just use

```
$ telnet localhost 80
GET / HTTP/1.0
```

for simple testing of Apache via HTTP, it's not so easy for HTTPS because of the SSL protocol between TCP and HTTP. With the help of OpenSSL's `s_client` command, however, you can do a similar check via HTTPS:

```
$ openssl s_client -connect localhost:443 -state -debug
GET / HTTP/1.0
```

Before the actual HTTP response you will receive detailed information about the SSL handshake. For a more general command line client which directly understands both HTTP and HTTPS, can perform GET and POST operations, can use a proxy, supports byte ranges, etc. you should have a look at the nifty [cURL](#) tool. Using this, you can check that Apache is responding correctly to requests via HTTP and HTTPS as follows:

```
$ curl http://localhost/
$ curl https://localhost/
```

## Why does the connection hang when I connect to my SSL-aware Apache server?

This can happen when you try to connect to a HTTPS server (or virtual server) via HTTP (eg, using `http://example.com/` instead of `https://example.com`). It can also happen when trying to connect via HTTPS to a HTTP server (eg, using `https://example.com/` on a server which doesn't support HTTPS, or which supports it on a non-standard port). Make sure

that you're connecting to a (virtual) server that supports SSL.

## Why do I get "Connection Refused" messages, when trying to access my newly installed Apache+mod\_ssl server via HTTPS?

This error can be caused by an incorrect configuration. Please make sure that your `Listen` directives match your `<VirtualHost>` directives. If all else fails, please start afresh, using the default configuration provided by `mod_ssl`.

## Why are the SSL\_XXX variables not available to my CGI & SSI scripts?

Please make sure you have `SSLOptions +StdEnvVars` enabled for the context of your CGI/SSI requests.

## How can I switch between HTTP and HTTPS in relative hyperlinks?

Usually, to switch between HTTP and HTTPS, you have to use fully-qualified hyperlinks (because you have to change the URL scheme). Using `mod_rewrite` however, you can manipulate relative hyperlinks, to achieve the same effect.

```
RewriteEngine on
RewriteRule    "^/(.*)_SSL$"    "https://%{SERVER_NAME}%1"
RewriteRule    "^/(.*)_NOSSL$"  "http://%{SERVER_NAME}%1"
```

This rewrite ruleset lets you use hyperlinks of the form `<a href="document.html_SSL">`, to switch to HTTPS in a relative link. (Replace SSL with NOSSL to switch to HTTP.)



- [What are RSA Private Keys, CSRs and Certificates?](#)
- [Is there a difference on startup between a non-SSL-aware Apache and an SSL-aware Apache?](#)
- [How do I create a self-signed SSL Certificate for testing purposes?](#)
- [How do I create a real SSL Certificate?](#)
- [How do I create and use my own Certificate Authority \(CA\)?](#)
- [How can I change the pass-phrase on my private key file?](#)
- [How can I get rid of the pass-phrase dialog at Apache startup time?](#)
- [How do I verify that a private key matches its Certificate?](#)
- [How can I convert a certificate from PEM to DER format?](#)
- [Why do browsers complain that they cannot verify my server certificate?](#)

## **What are RSA Private Keys, CSRs and Certificates?**

An RSA private key file is a digital file that you can use to decrypt messages sent to you. It has a public component which you distribute (via your Certificate file) which allows people to encrypt those messages to you.

A Certificate Signing Request (CSR) is a digital file which contains your public key and your name. You send the CSR to a Certifying Authority (CA), who will convert it into a real Certificate, by signing it.

A Certificate contains your RSA public key, your name, the name of the CA, and is digitally signed by the CA. Browsers that know the CA can verify the signature on that Certificate, thereby obtaining your RSA public key. That enables them to send messages which only you can decrypt.

See the [Introduction](#) chapter for a general description of the SSL

protocol.

## Is there a difference on startup between a non-SSL-aware Apache and an SSL-aware Apache?

Yes. In general, starting Apache with `mod_ssl` built-in is just like starting Apache without it. However, if you have a passphrase on your SSL private key file, a startup dialog will pop up which asks you to enter the pass phrase.

Having to manually enter the passphrase when starting the server can be problematic - for example, when starting the server from the system boot scripts. In this case, you can follow the steps [below](#) to remove the passphrase from your private key. Bear in mind that doing so brings additional security risks - proceed with caution!

## How do I create a self-signed SSL Certificate for testing purposes?

1. Make sure OpenSSL is installed and in your PATH.
2. Run the following command, to create `server.key` and `server.crt` files:

```
$ openssl req -new -x509 -nodes -out  
server.crt -keyout server.key
```

These can be used as follows in your `httpd.conf` file:

```
SSLCertificateFile    "/path/to/this/ser  
SSLCertificateKeyFile "/path/to/this/ser
```

3. It is important that you are aware that this `server.key` does *not* have any passphrase. To add a passphrase to the key,

you should run the following command, and enter & verify the passphrase as requested.

```
$ openssl rsa -des3 -in server.key -out  
server.key.new  
$ mv server.key.new server.key
```

Please backup the server . key file, and the passphrase you entered, in a secure location.

## **How do I create a real SSL Certificate?**

Here is a step-by-step description:

1. Make sure OpenSSL is installed and in your PATH.
2. Create a RSA private key for your Apache server (will be Triple-DES encrypted and PEM formatted):

```
$ openssl genrsa -des3 -out server.key 2048
```

Please backup this server . key file and the pass-phrase you entered in a secure location. You can see the details of this RSA private key by using the command:

```
$ openssl rsa -noout -text -in server.key
```

If necessary, you can also create a decrypted PEM version (not recommended) of this RSA private key with:

```
$ openssl rsa -in server.key -out  
server.key.unsecure
```

3. Create a Certificate Signing Request (CSR) with the server RSA private key (output will be PEM formatted):

```
$ openssl req -new -key server.key -out  
server.csr
```

Make sure you enter the FQDN ("Fully Qualified Domain Name") of the server when OpenSSL prompts you for the "CommonName", i.e. when you generate a CSR for a website which will be later accessed via `https://www.foo.dom/`, enter "www.foo.dom" here. You can see the details of this CSR by using

```
$ openssl req -noout -text -in server.csr
```

4. You now have to send this Certificate Signing Request (CSR) to a Certifying Authority (CA) to be signed. Once the CSR has been signed, you will have a real Certificate, which can be used by Apache. You can have a CSR signed by a commercial CA, or you can create your own CA to sign it. Commercial CAs usually ask you to post the CSR into a web form, pay for the signing, and then send a signed Certificate, which you can store in a `server.crt` file. For details on how to create your own CA, and use this to sign a CSR, see [below](#). Once your CSR has been signed, you can see the details of the Certificate as follows:

```
$ openssl x509 -noout -text -in server.crt
```

5. You should now have two files: `server.key` and `server.crt`. These can be used as follows in your `httpd.conf` file:

```
SSLCertificateFile    "/path/to/this/ser  
SSLCertificateKeyFile "/path/to/this/ser
```



The server .csr file is no longer needed.

## How do I create and use my own Certificate Authority (CA)?

The short answer is to use the CA.sh or CA.pl script provided by OpenSSL. Unless you have a good reason not to, you should use these for preference. If you cannot, you can create a self-signed certificate as follows:

1. Create a RSA private key for your server (will be Triple-DES encrypted and PEM formatted):

```
$ openssl genrsa -des3 -out server.key 2048
```

Please backup this server .key file and the pass-phrase you entered in a secure location. You can see the details of this RSA private key by using the command:

```
$ openssl rsa -noout -text -in server.key
```

If necessary, you can also create a decrypted PEM version (not recommended) of this RSA private key with:

```
$ openssl rsa -in server.key -out  
server.key.unsecure
```

2. Create a self-signed certificate (X509 structure) with the RSA key you just created (output will be PEM formatted):

```
$ openssl req -new -x509 -nodes -sha1 -days  
365 -key server.key -out server.crt -
```

```
extensions usr_cert
```

This signs the server CSR and results in a `server.crt` file. You can see the details of this Certificate using:

```
$ openssl x509 -noout -text -in server.crt
```

## **How can I change the pass-phrase on my private key file?**

You simply have to read it with the old pass-phrase and write it again, specifying the new pass-phrase. You can accomplish this with the following commands:

```
$ openssl rsa -des3 -in server.key -out  
server.key.new  
$ mv server.key.new server.key
```

The first time you're asked for a PEM pass-phrase, you should enter the old pass-phrase. After that, you'll be asked again to enter a pass-phrase - this time, use the new pass-phrase. If you are asked to verify the pass-phrase, you'll need to enter the new pass-phrase a second time.

## **How can I get rid of the pass-phrase dialog at Apache startup time?**

The reason this dialog pops up at startup and every re-start is that the RSA private key inside your `server.key` file is stored in encrypted format for security reasons. The pass-phrase is needed to decrypt this file, so it can be read and parsed. Removing the pass-phrase removes a layer of security from your server - proceed with caution!

1. Remove the encryption from the RSA private key (while keeping a backup copy of the original file):

```
$ cp server.key server.key.org  
$ openssl rsa -in server.key.org -out  
server.key
```

2. Make sure the server.key file is only readable by root:

```
$ chmod 400 server.key
```

Now server . key contains an unencrypted copy of the key. If you point your server at this file, it will not prompt you for a passphrase. HOWEVER, if anyone gets this key they will be able to impersonate you on the net. PLEASE make sure that the permissions on this file are such that only root or the web server user can read it (preferably get your web server to start as root but run as another user, and have the key readable only by root).

As an alternative approach you can use the ``SSLPassPhraseDialog exec:/path/to/program" facility. Bear in mind that this is neither more nor less secure, of course.

## **How do I verify that a private key matches its Certificate?**

A private key contains a series of numbers. Two of these numbers form the "public key", the others are part of the "private key". The "public key" bits are included when you generate a CSR, and subsequently form part of the associated Certificate.

To check that the public key in your Certificate matches the public portion of your private key, you simply need to compare these

numbers. To view the Certificate and the key run the commands:

```
$ openssl x509 -noout -text -in server.crt  
$ openssl rsa -noout -text -in server.key
```

The 'modulus' and the 'public exponent' portions in the key and the Certificate must match. As the public exponent is usually 65537 and it's difficult to visually check that the long modulus numbers are the same, you can use the following approach:

```
$ openssl x509 -noout -modulus -in server.crt |  
openssl md5  
$ openssl rsa -noout -modulus -in server.key |  
openssl md5
```

This leaves you with two rather shorter numbers to compare. It is, in theory, possible that these numbers may be the same, without the modulus numbers being the same, but the chances of this are overwhelmingly remote.

Should you wish to check to which key or certificate a particular CSR belongs you can perform the same calculation on the CSR as follows:

```
$ openssl req -noout -modulus -in server.csr |  
openssl md5
```

## **How can I convert a certificate from PEM to DER format?**

The default certificate format for OpenSSL is PEM, which is simply Base64 encoded DER, with header and footer lines. For some applications (e.g. Microsoft Internet Explorer) you need the certificate in plain DER format. You can convert a PEM file `cert.pem` into the corresponding DER file `cert.der` using the following command: **\$ openssl x509 -in cert.pem -out**

`cert.der -outform DER`

## Why do browsers complain that they cannot verify my server certificate?

One reason this might happen is because your server certificate is signed by an intermediate CA. Various CAs, such as Verisign or Thawte, have started signing certificates not with their root certificate but with intermediate certificates.

Intermediate CA certificates lie between the root CA certificate (which is installed in the browsers) and the server certificate (which you installed on the server). In order for the browser to be able to traverse and verify the trust chain from the server certificate to the root certificate it needs need to be given the intermediate certificates. The CAs should be able to provide you such intermediate certificate packages that can be installed on the server.

You need to include those intermediate certificates with the [SSLCertificateChainFile](#) directive.



- [Why do I get lots of random SSL protocol errors under heavy server load?](#)
- [Why does my webserver have a higher load, now that it serves SSL encrypted traffic?](#)
- [Why do HTTPS connections to my server sometimes take up to 30 seconds to establish a connection?](#)
- [What SSL Ciphers are supported by mod\\_ssl?](#)
- [Why do I get "no shared cipher" errors, when trying to use Anonymous Diffie-Hellman \(ADH\) ciphers?](#)
- [Why do I get a 'no shared ciphers' error when connecting to my newly installed server?](#)
- [Why can't I use SSL with name-based/non-IP-based virtual hosts?](#)
- [Is it possible to use Name-Based Virtual Hosting to identify different SSL virtual hosts?](#)
- [How do I get SSL compression working?](#)
- [When I use Basic Authentication over HTTPS the lock icon in Netscape browsers stays unlocked when the dialog pops up. Does this mean the username/password is being sent unencrypted?](#)
- [Why do I get I/O errors when connecting via HTTPS to an Apache+mod\\_ssl server with Microsoft Internet Explorer \(MSIE\)?](#)
- [How do I enable TLS-SRP?](#)
- [Why do I get handshake failures with Java-based clients when using a certificate with more than 1024 bits?](#)

## **Why do I get lots of random SSL protocol errors under heavy server load?**

There can be a number of reasons for this, but the main one is problems with the SSL session Cache specified by the [SSLSessionCache](#) directive. The DBM session cache is the most

likely source of the problem, so using the SHM session cache (or no cache at all) may help.

## **Why does my webserver have a higher load, now that it serves SSL encrypted traffic?**

SSL uses strong cryptographic encryption, which necessitates a lot of number crunching. When you request a webpage via HTTPS, everything (even the images) is encrypted before it is transferred. So increased HTTPS traffic leads to load increases.

## **Why do HTTPS connections to my server sometimes take up to 30 seconds to establish a connection?**

This is usually caused by a `/dev/random` device for [SSLRandomSeed](#) which blocks the `read(2)` call until enough entropy is available to service the request. More information is available in the reference manual for the [SSLRandomSeed](#) directive.

## **What SSL Ciphers are supported by mod\_ssl?**

Usually, any SSL ciphers supported by the version of OpenSSL in use, are also supported by [mod\\_ssl](#). Which ciphers are available can depend on the way you built OpenSSL. Typically, at least the following ciphers are supported:

1. RC4 with SHA1
2. AES with SHA1
3. Triple-DES with SHA1

To determine the actual list of ciphers available, you should run the following:

```
$ openssl ciphers -v
```

## Why do I get "no shared cipher" errors, when trying to use Anonymous Diffie-Hellman (ADH) ciphers?

By default, OpenSSL does *not* allow ADH ciphers, for security reasons. Please be sure you are aware of the potential side-effects if you choose to enable these ciphers.

In order to use Anonymous Diffie-Hellman (ADH) ciphers, you must build OpenSSL with `"-DSSL_ALLOW_ADH"`, and then add `"ADH"` into your [SSLCipherSuite](#).

## Why do I get a 'no shared ciphers' error when connecting to my newly installed server?

Either you have made a mistake with your [SSLCipherSuite](#) directive (compare it with the pre-configured example in `extra/httpd-ssl.conf`) or you chose to use DSA/DH algorithms instead of RSA when you generated your private key and ignored or overlooked the warnings. If you have chosen DSA/DH, then your server cannot communicate using RSA-based SSL ciphers (at least until you configure an additional RSA-based certificate/key pair). Modern browsers like NS or IE can only communicate over SSL using RSA ciphers. The result is the "no shared ciphers" error. To fix this, regenerate your server certificate/key pair, using the RSA algorithm.

## Why can't I use SSL with name-based/non-IP-based virtual hosts?

The reason is very technical, and a somewhat "chicken and egg" problem. The SSL protocol layer stays below the HTTP protocol layer and encapsulates HTTP. When an SSL connection (HTTPS) is established Apache/mod\_ssl has to negotiate the SSL protocol parameters with the client. For this, mod\_ssl has to consult the configuration of the virtual server (for instance it has to look for the cipher suite, the server certificate, etc.). But in order to go to the

correct virtual server Apache has to know the Host HTTP header field. To do this, the HTTP request header has to be read. This cannot be done before the SSL handshake is finished, but the information is needed in order to complete the SSL handshake phase. See the next question for how to circumvent this issue.

Note that if you have a wildcard SSL certificate, or a certificate that has multiple hostnames on it using subjectAltName fields, you can use SSL on name-based virtual hosts without further workarounds.

### **Is it possible to use Name-Based Virtual Hosting to identify different SSL virtual hosts?**

Name-Based Virtual Hosting is a very popular method of identifying different virtual hosts. It allows you to use the same IP address and the same port number for many different sites. When people move on to SSL, it seems natural to assume that the same method can be used to have lots of different SSL virtual hosts on the same server.

It is possible, but only if using a 2.2.12 or later web server, built with 0.9.8j or later OpenSSL. This is because it requires a feature that only the most recent revisions of the SSL specification added, called Server Name Indication (SNI).

Note that if you have a wildcard SSL certificate, or a certificate that has multiple hostnames on it using subjectAltName fields, you can use SSL on name-based virtual hosts without further workarounds.

The reason is that the SSL protocol is a separate layer which encapsulates the HTTP protocol. So the SSL session is a separate transaction, that takes place before the HTTP session has begun. The server receives an SSL request on IP address X and port Y (usually 443). Since the SSL request did not contain any Host: field, the server had no way to decide which SSL virtual host to

use. Usually, it just used the first one it found which matched the port and IP address specified.

If you are using a version of the web server and OpenSSL that support SNI, though, and the client's browser also supports SNI, then the hostname is included in the original SSL request, and the web server can select the correct SSL virtual host.

You can, of course, use Name-Based Virtual Hosting to identify many non-SSL virtual hosts (all on port 80, for example) and then have a single SSL virtual host (on port 443). But if you do this, you must make sure to put the non-SSL port number on the NameVirtualHost directive, e.g.

```
NameVirtualHost 192.168.1.1:80
```

Other workaround solutions include:

Using separate IP addresses for different SSL hosts. Using different port numbers for different SSL hosts.

## **How do I get SSL compression working?**

Although SSL compression negotiation was defined in the specification of SSLv2 and TLS, it took until May 2004 for RFC 3749 to define DEFLATE as a negotiable standard compression method.

OpenSSL 0.9.8 started to support this by default when compiled with the `zlib` option. If both the client and the server support compression, it will be used. However, most clients still try to initially connect with an SSLv2 Hello. As SSLv2 did not include an array of preferred compression algorithms in its handshake, compression cannot be negotiated with these clients. If the client disables support for SSLv2, either an SSLv3 or TLS Hello may be

sent, depending on which SSL library is used, and compression may be set up. You can verify whether clients make use of SSL compression by logging the `%{SSL_COMPRESS_METHOD}x` variable.

### **When I use Basic Authentication over HTTPS the lock icon in Netscape browsers stays unlocked when the dialog pops up. Does this mean the username/password is being sent unencrypted?**

No, the username/password is transmitted encrypted. The icon in Netscape browsers is not actually synchronized with the SSL/TLS layer. It only toggles to the locked state when the first part of the actual webpage data is transferred, which may confuse people. The Basic Authentication facility is part of the HTTP layer, which is above the SSL/TLS layer in HTTPS. Before any HTTP data communication takes place in HTTPS, the SSL/TLS layer has already completed its handshake phase, and switched to encrypted communication. So don't be confused by this icon.

### **Why do I get I/O errors when connecting via HTTPS to an Apache+mod\_ssl server with older versions of Microsoft Internet Explorer (MSIE)?**

The first reason is that the SSL implementation in some MSIE versions has some subtle bugs related to the HTTP keep-alive facility and the SSL close notify alerts on socket connection close. Additionally the interaction between SSL and HTTP/1.1 features are problematic in some MSIE versions. You can work around these problems by forcing Apache not to use HTTP/1.1, keep-alive connections or send the SSL close notify messages to MSIE clients. This can be done by using the following directive in your SSL-aware virtual host section:

```
SetEnvIf User-Agent "MSIE [2-5]" \
```

```
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
```

Further, some MSIE versions have problems with particular ciphers. Unfortunately, it is not possible to implement a MSIE-specific workaround for this, because the ciphers are needed as early as the SSL handshake phase. So a MSIE-specific `SetEnvIf` won't solve these problems. Instead, you will have to make more drastic adjustments to the global parameters. Before you decide to do this, make sure your clients really have problems. If not, do not make these changes - they will affect *all* your clients, MSIE or otherwise.

## How do I enable TLS-SRP?

TLS-SRP (Secure Remote Password key exchange for TLS, specified in RFC 5054) can supplement or replace certificates in authenticating an SSL connection. To use TLS-SRP, set the `SSLSRPVerifierFile` directive to point to an OpenSSL SRP verifier file. To create the verifier file, use the `openssl` tool:

```
openssl srp -srpvfile passwd.srpv -add username
```

After creating this file, specify it in the SSL server configuration:

```
SSLSRPVerifierFile /path/to/passwd.srpv
```

To force clients to use non-certificate TLS-SRP cipher suites, use the following directive:

```
SSLCipherSuite "!DSS:!aRSA:SRP"
```

## Why do I get handshake failures with Java-based

## clients when using a certificate with more than 1024 bits?

Beginning with version 2.4.7, `mod_ssl` will use DH parameters which include primes with lengths of more than 1024 bits. Java 7 and earlier limit their support for DH prime sizes to a maximum of 1024 bits, however.

If your Java-based client aborts with exceptions such as `java.lang.RuntimeException: Could not generate DH keypair` and `java.security.InvalidAlgorithmParameterException: Prime size must be multiple of 64, and can only range from 512 to 1024 (inclusive)`, and `httpd` logs `tlsv1 alert internal error (SSL alert number 80)` (at `LogLevel` info or higher), you can either rearrange `mod_ssl`'s cipher list with `SSLCipherSuite` (possibly in conjunction with `SSLHonorCipherOrder`), or you can use custom DH parameters with a 1024-bit prime, which will always have precedence over any of the built-in DH parameters.

To generate custom DH parameters, use the `openssl dhparam 1024` command. Alternatively, you can use the following standard 1024-bit DH parameters from [RFC 2409](#), section 6.2:

```
-----BEGIN DH PARAMETERS-----
MIGHAoGBAP/////////yQ/aoiFowjTExmKLGnWc0SkCTgiKZ8x0Agu+pjsTmyJR
Sgh5jjQE3e+VgbPN0kMbMCsKbfJfFDdP4TVtbVHCREsFtXZiXn7G9ExC6aY37WsL
/1y29Aa37e44a/taiZ+lrp8kEXxLH+ZJKGZR70ZTgf/////////AgEC
-----END DH PARAMETERS-----
```

Add the custom parameters including the "BEGIN DH PARAMETERS" and "END DH PARAMETERS" lines to the end of the first certificate file you have configured using the `SSLCertificateFile` directive.



- [What information resources are available in case of mod\\_ssl problems?](#)
- [What support contacts are available in case of mod\\_ssl problems?](#)
- [What information should I provide when writing a bug report?](#)
- [I had a core dump, can you help me?](#)
- [How do I get a backtrace, to help find the reason for my core dump?](#)

## What information resources are available in case of mod\_ssl problems?

The following information resources are available. In case of problems you should search here first.

### Answers in the User Manual's F.A.Q. List (this)

[http://httpd.apache.org/docs/2.4/ssl/ssl\\_faq.html](http://httpd.apache.org/docs/2.4/ssl/ssl_faq.html)

First check the F.A.Q. (this text). If your problem is a common one, it may have been answered several times before, and been included in this doc.

## What support contacts are available in case of mod\_ssl problems?

The following lists all support possibilities for mod\_ssl, in order of preference. Please go through these possibilities *in this order* - don't just pick the one you like the look of.

1. *Send a Problem Report to the Apache httpd Users Support Mailing List*

[users@httpd.apache.org](mailto:users@httpd.apache.org)

This is the second way of submitting your problem report. Again, you must subscribe to the list first, but you can then easily discuss your problem with the whole Apache httpd user

community.

## 2. *Write a Problem Report in the Bug Database*

[http://httpd.apache.org/bug\\_report.html](http://httpd.apache.org/bug_report.html)

This is the last way of submitting your problem report. You should only do this if you've already posted to the mailing lists, and had no success. Please follow the instructions on the above page *carefully*.

## **What information should I provide when writing a bug report?**

You should always provide at least the following information:

### **Apache httpd and OpenSSL version information**

The Apache version can be determined by running `httpd -v`. The OpenSSL version can be determined by running `openssl version`. Alternatively, if you have Lynx installed, you can run the command `lynx -mime_header http://localhost/ | grep Server` to gather this information in a single step.

### **The details on how you built and installed Apache httpd and OpenSSL**

For this you can provide a logfile of your terminal session which shows the configuration and install steps. If this is not possible, you should at least provide the [configure](#) command line you used.

### **In case of core dumps please include a Backtrace**

If your Apache httpd dumps its core, please attach a stack-frame ```backtrace"` (see [below](#) for information on how to get this). This information is required in order to find a reason for your core dump.

### **A detailed description of your problem**

Don't laugh, we really mean it! Many problem reports don't

include a description of what the actual problem is. Without this, it's very difficult for anyone to help you. So, it's in your own interest (you want the problem be solved, don't you?) to include as much detail as possible, please. Of course, you should still include all the essentials above too.

## **I had a core dump, can you help me?**

In general no, at least not unless you provide more details about the code location where Apache dumped core. What is usually always required in order to help you is a backtrace (see next question). Without this information it is mostly impossible to find the problem and help you in fixing it.

## **How do I get a backtrace, to help find the reason for my core dump?**

Following are the steps you will need to complete, to get a backtrace:

1. Make sure you have debugging symbols available, at least in Apache. On platforms where you use GCC/GDB, you will have to build Apache+mod\_ssl with ```OPTIM="-g -ggdb3""` to get this. On other platforms at least ```OPTIM="-g""` is needed.
2. Start the server and try to reproduce the core-dump. For this you may want to use a directive like ```CoreDumpDirectory /tmp"` to make sure that the core-dump file can be written. This should result in a `/tmp/core` or `/tmp/httpd.core` file. If you don't get one of these, try running your server under a non-root UID. Many modern kernels do not allow a process to dump core after it has done a `setuid( )` (unless it does an `exec( )`) for security reasons (there can be privileged information left over in memory). If necessary, you can run

`/path/to/httpd -X` manually to force Apache to not fork.

3. Analyze the core-dump. For this, run `gdb /path/to/httpd /tmp/httpd.core` or a similar command. In GDB, all you have to do then is to enter `bt`, and voila, you get the backtrace. For other debuggers consult your local debugger manual.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [How-To / Tutorials](#)

# (Authentication), (Authorization), (Access Control)

.

.

(authentication)

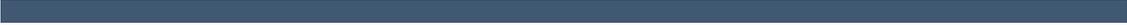
. (authorization)

.



<u>mod auth basic</u>	<u>Allow</u>
<u>mod authn file</u>	<u>AuthGroupFile</u>
<u>mod authz groupfile</u>	<u>AuthName</u>
<u>mod authz host</u>	<u>AuthType</u>
	<u>AuthUserFile</u>
	<u>Deny</u>
	<u>Options</u>
	<u>Require</u>





,

.

'''

.



```
( <Directory> ) (
```

```
.htaccess
```

```
AllowOverride .
```

```
, AllowOverride .
```

```
AllowOverride AuthConfig
```

```
, .  
.  
.  
.
```



```
, /usr/local/apache/htdocs ()  
/usr/local/apache/passwd .
```

[htpasswd](#)

```
htpasswd -c /usr/local/apache/passwd/passwords rbowen
```

htpasswd ,

```
# htpasswd -c /usr/local/apache/passwd/passwords rbowen  
New password: mypassword  
Re-type new password: mypassword  
Adding password for user rbowen
```

```
htpasswd  
/usr/local/apache/bin/htpasswd
```

```
.htaccess . ,  
/usr/local/apache/htdocs/secret ,  
/usr/local/apache/htdocs/secret/.htaccess  
httpd.conf <Directory  
/usr/local/apache/apache/htdocs/secret>
```

```
AuthType Basic  
AuthName "Restricted Files"  
AuthUserFile /usr/local/apache/passwd/passwords  
Require user rbowen
```

[AuthType](#)  
[mod\\_auth\\_basic](#) . Basic



```
( rbowen) .  
AuthGroupFile .
```

```
GroupName: rbowen dpitts sungo rshersey
```

```
htpasswd /usr/local/apache/passwd/passwords dpitts
```

```
, .( -c ).
```

```
.htaccess .
```

```
AuthType Basic  
AuthName "By Invitation Only"  
AuthUserFile /usr/local/apache/passwd/passwords  
AuthGroupFile /usr/local/apache/passwd/groups  
Require group GroupName
```

```
GroupName password .
```

```
Require valid-user
```

```
Require user rbowen
```

```
(
```



---

Basic

( )

.

.

.

.

.

.

,

.



## Allow Deny

```
Allow from address
```

```
address IP ( IP ) ( ).
```

```
Deny from 205.252.46.165
```

. IP

```
Deny from host.example.com
```

```
Deny from 192.101.205  
Deny from cyberthugs.com moreidiots.com  
Deny from ke
```

## Order Deny Allow

```
Order deny,allow  
Deny from all  
Allow from dev.example.com
```

## Allow ,



## mod\_auth\_basic mod\_authz\_host

.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [How-To / Tutorials](#)

**: CGI**



```
mod alias AddHandler
mod cgi Options
ScriptAlias
```

CGI (Common Gateway Interface) CGI CGI ,
( )
CGI , CGI .



CGI CGI . . .

## ScriptAlias

ScriptAlias CGI .  
CGI .

ScriptAlias .

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

```
httpd.conf . ScriptAli  
Alias URL . Alias  
DocumentRoot . Alias ScriptAli  
ScriptAlias URL CGI .  
/cgi-bin/ /usr/local/apache2  
CGI .
```

```
, URL http://www.example.com/cgi-bin/test.pl  
/usr/local/apache2/cgi-bin/test.pl .
```

## ScriptAlias CGI

```
CGI ScriptAlias . CGI  
CGI  
, UserDir  
cgi-bin , CGI .
```

```
CGI ., AddHandle  
cgi-script ., Options Exec
```

## Options CGI

## Options

CGI .

```
<Directory /usr/local/apache2/htdocs/somedir>
  Options +ExecCGI
</Directory>
```

## CGI . AddHandler

CGI .  
cgi pl CGI .

```
AddHandler cgi-script .cgi .pl
```

## **.htaccess**

.htaccess httpd.conf CGI

.

.cgi CGI .

```
<Directory /home/*/public_html>
  Options +ExecCGI
  AddHandler cgi-script .cgi
</Directory>
```

cgi-bin CGI .

```
<Directory /home/*/public_html/cgi-bin>
  Options ExecCGI
  SetHandler cgi-script
</Directory>
```



```
CGI .
CGI MIME-type . HTTP
```

```
Content-type: text/html
```

```
HTML . HTML ,
gif HTML CGI .
```

```
CGI .
```

```
CGI
```

```
CGI . first.pl ,
```

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello, World.";
```

```
Perl . ( )
/usr/bin/perl .
content-type carriage-return . HTTP
, . "Hello, World." . .
```

```
http://www.example.com/cgi-bin/first.pl
```

```
, Hello, World. . ,
```



CGI

**CGI**

! . , CGI  
Content-Type .

**CGI "POST Method Not Allowed"**

CGI

**"Forbidden"**

**"Internal Server Error"**

CGI "Premature end of  
headers" . CGI  
HTTP . CGI

www).

```
chmod a+x first.pl
```

CGI

PATH . ( ,

CGI

(

PATH

```
#!/usr/bin/perl
```

```
.  
, CGI .
```

```
CGI .  
. . ,
```

```
cd /usr/local/apache2/cgi-bin  
./first.pl
```

```
(perl .  
Content-Type HTTP .  
Premature end of script  
CGI .
```

```
. . .  
, . ,  
.
```

## Suexec

```
suexec  
. Suexec , CGI  
Premature end of script headers.
```

```
suexec apachectl -V SUEXEC_BIN .  
suexec , suexec .
```

```
suexec . suexec
```

suexec ( ) .  
, suexec -V suexec

[suexec](#)



CGI

"Hello, World."

), , . path (

CGI (Netscape, IE, Lynx), (, IIS, WebSite), CGI

CGI , - . <http://hoohoo.ncsa.uiuc.edu/cgi/env.html> .

Perl CGI

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
foreach $key (keys %ENV) {
    print "$key --> $ENV{$key}<br>";
}
```

## STDIN STDOUT

, ( STDIN) ( STDOUT) . STDIN  
, STDOUT .

CGI (form) POST CGI

" " . (=) , (&)

., , 16 .

name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Squirrel%20Mor

URL . QUERY\_STRING  
GET . FORM METHOD HTML (form)  
POST .

. CGI



---

CGI

Perl CGI

[CPAN](#)

. CGI

CGI.pm.

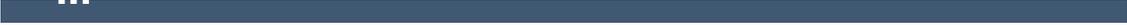
CGI::Lite

C CGI .

<http://www.bout>

CGIC .





CGI . [comp.infosystems.www.authoring.cgi](http://comp.infosystems.www.authoring.cgi)

CGI . HTML Writers Guild -servers

<http://www.hwg.org/lists/hwg-servers/>

.

CGI

CGI .

[Common Gateway Interface RFC](#) .

CGI

, , CGI

CGI



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [How-To / Tutorials](#)

## : Server Side Includes

Server-side includes HTML .



```
mod_include Options
mod_cgi XBitHack
mod_expires AddType
           SetOutputFilter
           BrowserMatchNoCase
```

SSI Server Side Includes .

SSI

HTML

SSI .

SSI .



---

SSI (Server Side Includes) HTML ,  
SSI CGI

HTM

SSI  
. SSI



```
SSI httpd.conf .htaccess .
```

```
Options +Includes
```

```
SSI . Options  
SSI
```

```
SSI . . .  
.shtml .
```

```
AddType text/html .shtml  
AddOutputFilter INCLUDES .shtml
```

```
SSI SSI
```

```
XBitHack .
```

```
XBitHack on
```

```
XBitHack SSI . SSI  
chmod .
```

```
chmod +x pagename.html
```

```
. .shtml .html SSI  
.  
SSI XBitHack .  
. . , .  
.  
SSI content length  
.
```

1. XBitHack Full . (include)

.

2. mod\_expires



SSI .

```
<!--#element attribute=value attribute=value ... -->
```

HTML SSI

HTML . SSI

element . .

SSI

```
<!--#echo var="DATE_LOCAL" -->
```

echo element .

CGI

set element

,  
attribute .

config element timefmt

```
<!--#config timefmt="%A %B %d, %Y" -->  
Today is <!--#echo var="DATE_LOCAL" -->
```

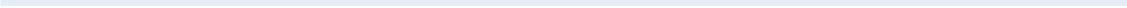
```
<!--#flastmod file="index.html" -->
```

element timefmt .

**CGI**

SSI , `` " CGI .

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```



HTML SSI .

?

SSI . . .  
HTML . SSI .

```
<!--#config timefmt="%A %B %d, %Y" -->  
<!--#flastmod file="ssi.shtml" --> ;
```

ssi.shtml .  
LAST\_MODIFIED .

```
<!--#config timefmt="%D" -->  
This file last modified <!--#echo var="LAST_MODIFIED" -->
```

timefmt strftime . .

(header) (footer) .  
include SSI . includ  
file attribute virtual attribute . file attribu  
., (/) ../ .  
virtual attribute . / ,

```
<!--#include virtual="/footer.html" -->
```

LAST\_MODIFIED .  
SSI , .



config()

config() .

SSI

[an error occurred while processing this directive]

config element errmsg attribute .

```
<!--#config errmsg="[It appears that you don't know how to use SSI]" -->
```

SSI

. (?)

sizefmt attribute

config() .

bytes, Kb Mb

abbrev .



```
CGI SSI . e
.SSI ( /bin/sh Win32
DOS) . , .
```

```
<pre>
<!--#exec cmd="ls" -->
</pre>
```

or, on Windows

```
<pre>
<!--#exec cmd="dir" -->
</pre>
```

```
dir ``<dir>" , .
exec . ""
, Options Include
SSI exec .
```



SSI , .

1.2

., 1.2

set . .

```
<!--#set var="name" value="Rich" -->
```

```
( , LAST_MODIFIED)  
($)
```

```
<!--#set var="modified" value="$LAST_MODIFIED" -->
```

```
<!--#set var="cost" value="\$100" -->
```

```
,  
)
```

```
<!--#set var="date" value="{DATE_LOCAL}_{DATE_GMT}" -->
```

SSI .

if, elif, else, endif .

```
<!--#if expr="test_condition" -->
<!--#elif expr="test_condition" -->
<!--#else -->
<!--#endif -->
```

*test\_condition* . , ``"  
.( .) , [mod\\_j](#)  
. .

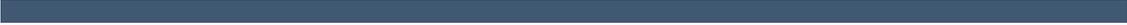
```
BrowserMatchNoCase macintosh Mac
BrowserMatchNoCase MSIE InternetExplorer
```

Internet Explorer ``Mac" ``InternetE  
.  
SSI .

```
<!--#if expr="${Mac} && ${InternetExplorer}" -->
<!--#else -->
  JavaScript
<!--#endif -->
```

IE . JavaScript  
IE .  
( )  
CGI .





SSI CGI

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [How-To / Tutorials](#)

**: .htaccess**

. .

.htaccess .



<u>core</u>	<u>AccessFileName</u>
<u>mod_authn_file</u>	<u>AllowOverride</u>
<u>mod_authz_groupfile</u>	<u>Options</u>
<u>mod_cgi</u>	<u>AddHandler</u>
<u>mod_include</u>	<u>SetHandler</u>
<u>mod_mime</u>	<u>AuthType</u>
	<u>AuthName</u>
	<u>AuthUserFile</u>
	<u>AuthGroupFile</u>
	<u>Require</u>



```
.htaccess (" ")
```

```
:
```

```
.htaccess , AccessFileName  
, .config .
```

```
AccessFileName .config
```

```
.htaccess . AllowOverride  
. .htaccess .  
, Override AllowOv  
. .  
, AddDefaultCharset .htaccess  
( . ) Override FileInfo .  
.htaccess AllowOverride FileInfo
```

```
:
```

```
: , , directory,  
.htaccess  
Override: FileInfo
```

```
.htaccess
```

```
".htaccess"
```



```

        .htaccess . ,
    .htaccess . .
    , .
    .htaccess root .htacce
    .
    . , ISP
    .
    .htaccess . .htaccess
    <Directory> .
    .htaccess .
    . AllowOverride .htaccess ,
    .htaccess . .htaccess
    !, .htaccess .
    .hta
    .) /www/htdocs/example ,
    .

```

```

/.htaccess
/www/.htaccess
/www/htdocs/.htaccess
/www/htdocs/example/.htaccess

```

4 .

```

.htaccess . .)
.
.
.
AllowOverride
/www/htdocs/example .htaccess

```

```
<Directory /www/htdocs/example> Directory
```

.

```
/www/htdocs/example .htaccess :
```

```
/www/htdocs/example .htaccess
```

```
AddType text/example .exm
```

### httpd.conf

```
<Directory /www/htdocs/example>  
  AddType text/example .exm  
</Directory>
```

```
AllowOverride none .htaccess .
```

```
AllowOverride None
```



```
.htaccess .htac
```

```
.htaccess . .
```

```
.htaccess .htaccess
```

```
.
```

```
:
```

```
/www/htdocs/example1 .htaccess .
```

```
Options +ExecCGI
```

```
(: .htaccess " Options "AllowOverride  
Options".)
```

```
/www/htdocs/example1/example2 .htacc
```

```
.
```

```
Options Includes
```

```
.htaccess Options Includes  
/www/htdocs/example1/example2 CGI .
```





`.htaccess`      Server Side Includes  
`.htaccess`      .

```
Options +Includes  
AddType text/html shtml  
AddHandler server-parsed shtml
```

```
AllowOverride Options AllowOverride  
FileInfo .
```

server-side includes      [SSI](#) .



.htaccess CGI ,

```
Options +ExecCGI
AddHandler cgi-script cgi pl
```

CGI .

```
Options +ExecCGI
SetHandler cgi-script
```

```
AllowOverride Options AllowOverride
FileInfo .
```

CGI [CGI](#) .



---

.htaccess

AllowOverride

AllowOverride None

None

.htaccess

.ht



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [How-To / Tutorials](#)



UserDir .  
http://example.com/~username/ " username"  
UserDir .



URL



---

```
mod userdir UserDir  
DirectoryMatch  
AllowOverride
```



UserDir . .

. ,

```
UserDir public_html
```

URL `http://example.com/~rbowen/file.html`  
`/home/rbowen/public_html/file.html` .

. ,

```
UserDir /var/html
```

URL `http://example.com/~rbowen/file.html`  
`/var/html/rbowen/file.html` .

(\*)

. , :

```
UserDir /var/www/*/docs
```

URL `http://example.com/~rbowen/file.html`  
`/var/www/rbowen/docs/file.html` .



UserDir

:

```
UserDir enabled
UserDir disabled root jro fish
```

disabled

.,

:

```
UserDir disabled
UserDir enabled rbowen krietz
```

UserDir .



## cgi

cgi-bin      <Directory>      cgi

```
<Directory /home/*/public_html/cgi-bin/>  
Options ExecCGI  
SetHandler cgi-script  
</Directory>
```

UserDir public\_html , cgi  
example.cgi .

```
http://example.com/~rbowen/cgi-bin/example.cgi
```



,  
[AllowOverride](#)

[.htaccess](#)

.htaccess .



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Platform Specific Notes](#)

# Microsoft Windows



Microsoft Windows 2.0 , , .

( )  
[Windows](#) .

## Microsoft Windows :

- **Windows NT:** Windows NT Windows .  
Windows NT, Windows 2000, Windows XP, Windows .Net Server 2003 .
- **Windows 9x:** Windows . Windows 95 (OSR2), Windows 98, Windows ME .



---

2.0 Windows Windows NT. Intel AM  
x86 . Windows 9x

TCP/IP . Windows 95 , Winsock  
. Windows 95 Winsock 2 .

Windows NT 4.0 4 TCP/IP Winsock ,  
6 .



<http://httpd.apache.org/download.cgi>

. ,  
.  
Windows .msi Windows .  
Microsoft . .zip . Mi  
C++ (Visual Studio) .



Microsoft Installer 1.2 . Windows 9x  
Installer 2.0 , Windows NT 4.0 2000  
. Windows XP .

2.0 . 1.3

. 2.0

.msi . :

1. **(Network Domain).** DNS . ,  
DNS server.mydomain.net mydomain.net
2. **(Server Name).** DNS .  
server.mydomain.net .
3. **(Administrator's Email Address).**  
.
4. **(For whom to install Apache) 80**  
for All Users, on Port 80, as a Service  
Recommended ( , 80 , service - ).  
service ( , ).  
only for the Current User, on Por  
8080, when started Manually ( , 8080 ,  
).
5. **(The installation type).**  
Typical . Custom .  
13 .
6. **(Where to install).** C:\Program  
Files\Apache Group, Apache2 .  
conf .

```
., .default . ,
conf\httpd.conf conf\httpd.conf.default
. .default , .
, htdocs\index.html
(index.html.default ).,
. , .
conf .
. . .
```



conf . , Windows

Windows :

- Windows , , 2.

MaxRequestsPerChild:

MaxRequestsPerChild 0

```
httpd.conf
```

ThreadsPerChild:

ThreadsPerChild 50.

- Windows .

- Windows  
 \Apache2\modules  
 LoadModule . , status  
 status ) :

```
LoadModule status_module modules/mod_status.so
```

- Microsoft IIS Windows ISAPI (Internet Application Programming Interface) (, )  
 . [ISAPI](#) .
- CGI [ScriptInterpreterSource](#)
- Windows .htaccess , [AccessFile](#)
- Windows NT Windows error.log . Windows  
 Windows NT 4.0 , Windows MMC

**Windows 9x Windows** .



Windows NT service . Windows

9x

```
service ."
" service " , service ."
. service Administrators
```

Apache Service Monitor .

```
. monitor service service (
)
```

bin

Windows NT service :

```
apache -k install
```

```
service .
```

```
apache -k install -n "MyServiceName"
```

```
service :
```

```
apache -k install -n "MyServiceName" -f "c:\files\my.conf"
```

```
-k install , service Apache2
conf\httpd.conf .
```

```
service .:
```

```
apache -k uninstall
```

```
service :
```

```
apache -k uninstall -n "MyServiceName"
```

service , , Apache Service Monitor NET S  
Apache2, NET STOP Apache2 Windows  
service :

```
apache -n "MyServiceName" -t
```

service . service :

```
apache -k start
```

service :

```
apache -k stop
```

```
apache -k shutdown
```

service :

```
apache -k restart
```

service (LocalSystem) .  
Windows LocalSystem , named pipes, DC  
secure RPC .

**LocalSystem !**

service .

.

1. .

2. Windows NT 4.0 User Manager for Domains , Windows 2000 XP " " MMC .
3. Users .
4. ( htdocs cgi-bin) .
5. logs (RWXD) .
6. Apache.exe (RX) .

```
service (RWXD) log
Apache2 (RX) .
```

```
" " " " ,
. service
```

```
Error code 2186 service ""
. , .
```

```
service Windows Service Control Manager .
, "" :
```

```
Could not start the Apache2 service on \\COMPUTER
Error 1067; The process terminated unexpectedly.
```

```
service .
```

```
Windows 9x Windows NT service .
```

```
. .
```

```
service :
```

- . ,

```
apache -n "MyServiceName" -k start
```

service . httpd.conf

- Windows 9x NET START NET STOP . service .

- Windows 9x . Windows 9x . Apache Software Foundation Windows 9x .

Windows NT service , , service , , Apache Service Monitor Windows 9x service .



```
service . (Windows 9x  
).
```

```
, :
```

```
apache
```

```
Control-C .
```

```
, --> --> Apache HTTP Server 2.0.xx --  
Control Apache Server Start Apache in Console
```

```
. . sE  
Control-C . ,  
service service . service .
```

```
:
```

```
apache -k shutdown
```

```
Control-C .
```

```
, . . . .
```

```
apache -k restart
```

```
: kill -TERM pid kill  
Windows. -k kill .
```

```
--> .
```

```
apache . logs ,
```

```
. :
```

```
c:  
cd "\\Program Files\\Apache Group\\Apache2\\bin"
```

```
apache
```

Control-C . :

```
cd ..\logs  
more < error.log
```

- -f :

```
apache -f "c:\my server files\anotherconfig.conf"
```

```
apache -f files\anotherconfig.conf
```

- -n service , service :

```
apache -n "MyServiceName"
```

ServerRoot .

```
-f -n , conf\httpd.conf  
-V SE  
:
```

```
apache -V
```

ServerRoot :

1. -C ServerRoot .
2. -d .
3. .

4. registry .
5. server root. /apache, apache -V  
HTTPD\_ROOT .

```

install
for all users HKEY_LOCAL_MACHINE
):

```

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Group\Apache\2.0.43
```

```

" " HKEY_CURRENT_USER .
:

```

```
HKEY_CURRENT_USER\SOFTWARE\Apache Group\Apache\2.0.43
```

```

ServerRoot , conf
httpd.conf . ServerRoot ,
httpd.conf ServerRoot .

```



```
( service ) ( Listen " "
) 80 . URL :
```

```
http://localhost/
```

```
error.log . DNS (Domain Name Service)
URL :
```

```
http://127.0.0.1/
```

```
conf . , Windows NT service
```

```
TCP/IP
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Platform Specific Notes](#)

# Microsoft Windows

.

. [Microsoft Windows](#) .





<http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe> .  
awk95.exe awk.exe .

```
Developer Studio IDE Tools Options... Directories  
(Developer Studio 7.0 Projects - VC++ Directories pane)  
Executable files awk.exe . awk.exe  
, PATH .
```

```
Cygwin ( http://www.cygwin.com/) gawk.exe awk  
, awk.exe gawk.exe . Windows  
InstallBin . cygwin  
gawk.exe awk.exe .
```

- `[] OpenSSL ( mod\_ssl ab.exe ssl )`

```
: .
```

```
Foundation OpenSSL OpenSSL , ,  
. . .
```

```
mod\_ssl (SSL ab.exe) abs , OpenSSL  
http://www.openssl.org/source/ srclib openssl  
. release debug 0.9.7  
, :
```

```
perl Configure VC-WIN32  
perl util\mkfiles.pl >MINFO  
perl util\mk1mf.pl dll no-asm no-mdc2 no-rc5 no-idea VC-  
WIN32 >makefile  
perl util\mk1mf.pl dll debug no-asm no-mdc2 no-rc5 no-idea  
VC-WIN32 >makefile.dbg  
perl util\mkdef.pl 32 libeay no-asm no-mdc2 no-rc5 no-idea  
>ms\libeay32.def  
perl util\mkdef.pl 32 ssleay no-asm no-mdc2 no-rc5 no-idea  
>ms\ssleay32.def  
nmake
```

```
nmake -f makefile.dbg
```

- [] zlib ( mod\_deflate )

```
Zlib srclib zlib ,  
      mod_deflate .
```

```
--      mod_deflate 1.1.4
```

Zlib <http://www.gzi>

.



---

.  
Makefile.win makefile . Windows NT rel  
debug :

```
nmake /f Makefile.win _apacher  
nmake /f Makefile.win _apached
```



VC++ Visual Studio . Visual  
Studio workspace Apache.dsw . workspace  
.dsp . ,

Apache.dsw workspace InstallBin ( Release Debug  
) Active Project . InstallBin ,  
dll Makefile.win . InstallBin Settings, General  
Build command line INSTDIR= . INSTDIR  
/Apache2 . () Build

.dsp Visual C++ 6.0 . Visual C++ 5.0 (97)  
. Visual C++ 7.0 (.net) Apache.dsw .dsp  
Apache.sln .msproj . .dsp  
! VC++ 7.0 IDE Apache.dsw .

, Visual C++ 7.0 (.net) Build , Configuration Manager  
Debug Release abs, [mod\\_ssl](#), [mod\\_deflate](#) Solution  
modules . srclib openssl zlib  
( ) IDE BinBuild

Export .mak , Visual C++ 5.0 [mod\\_ssl](#),  
ab), [mod\\_deflate](#) . VC++ 7.0 (.net)  
nmake . VC++ 5.0 6.0 IDE , Project  
Export for all makefiles .

```
perl srclib\apr\build\fixwin32mak.pl
```

httpd .

, Visual Studio 6.0 . ,  
7.0 .



---

Apache.dsw workspace makefile.win nmake

.dsp :

1. srclib\apr\apr.dsp
2. srclib\apr\libapr.dsp
3. srclib\apr-util\uri\gen\_uri\_delims.dsp
4. srclib\apr-util\xml\expat\lib\xml.dsp
5. srclib\apr-util\aprutil.dsp
6. srclib\apr-util\libaprutil.dsp
7. srclib\pcre\dftables.dsp
8. srclib\pcre\pcre.dsp
9. srclib\pcre\pcreposix.dsp
10. server\gen\_test\_char.dsp
11. libhttpd.dsp
12. Apache.dsp

, modules\ .

support\

Windows support\win32\ .

1. support\ab.dsp
2. support\htdigest.dsp
3. support\htpasswd.dsp
4. support\logresolve.dsp
5. support\rotatelogs.dsp

6. support\win32\ApacheMonitor.dsp

7. support\win32\wintty.dsp

server root .

*dir* nmake :

```
nmake /f Makefile.win installr INSTDIR=dir
```

```
nmake /f Makefile.win installd INSTDIR=dir
```

INSTDIR *dir* . \Apache2 .

:

- *dir*\bin\Apache.exe -
- *dir*\bin\ApacheMonitor.exe -
- *dir*\bin\htdigest.exe - Digest auth
- *dir*\bin\htdbm.exe - SDBM auth
- *dir*\bin\htpasswd.exe - Basic auth
- *dir*\bin\logresolve.exe - dns
- *dir*\bin\rotatelogs.exe -
- *dir*\bin\wintty.exe -
- *dir*\bin\libapr.dll - Apache Portable Runtime
- *dir*\bin\libaprutil.dll - Apache Utility Runtime
- *dir*\bin\libhttpd.dll - Apache Core
- *dir*\modules\mod\_\*.so -
- *dir*\conf -
- *dir*\logs -
- *dir*\include - C
- *dir*\lib -

```
.dsp release . .mak .
NMAKE .dsp .
export . Microsoft Developer Studio .
```

```
, makefile export BuildBin ( _apacher
_apached ) .
.
```

```
.mak .mak ( .dep) Platform SDK .
DevStudio\SharedIDE\bin\ (VC5)
DevStudio\Common\MSDev98\bin\ (VC6)
sysincl.dat . (sys/time.h
sys\time.h, ).
.
srclib/apr/build/fixwin32mak.pl .mak
.
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Platform Specific Notes](#)

# Novell NetWare

Novell NetWare 6.0 2.0 ,

dev-httpd

[\(FAQ\)](#) ,

, NetWare

[novell.devsup.webserver](http://novell.devsup.webserver)

( )

[NetWare](#)



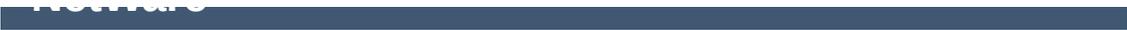
---

2.0 NetWare 6.0 service pack 3 . SP3 service  
pack [NetWare Libraries for C \(LibC\)](#) .

NetWare service pack .

service pack [NetWare Libraries for C \(LibC\)](#)  
NetWare 5.1 NetWare 2.0 . : NetWare  
2.0 .





<http://www.apache.org/> ( ) .

/ , ftp

. NetWare

.



NetWare . NetWare 2.0

NetWare ( sys:/ap

- SYS: ( )
- httpd.conf ServerRoot ServerName
- 

SEARCH ADD SYS:\APACHE2

SYS:/APACHE2

NetWare ( sys:  
):

- NetWare Apache2
- APACHE2.NLM APRLIB.NLM SYS:/APACHE2
- SYS:/APACHE2 BIN
- HTDIGEST.NLM, HTPASSWD.NLM, HTDBM.NLM,  
LOGRES.NLM, ROTLOGS.NLM SYS:/APACHE2/BIN
- SYS:/APACHE2 CONF
- HTTPD-STD.CONF SYS:/APACHE2/CONF  
HTTPD.CONF
- MIME.TYPES, CHARSET.CONV, MAGIC  
SYS:/APACHE2/CONF
- \HTTPD-2.0\DOCS\ICONS  
SYS:/APACHE2/ICONS
- \HTTPD-2.0\DOCS\MANUAL  
SYS:/APACHE2/MANUAL
- \HTTPD-2.0\DOCS\ERROR  
SYS:/APACHE2/ERROR

- \HTTPD-2.0\DOCS\DICROOT  
SYS:/APACHE2/HTDOCS
- SYS:/APACHE2/LOGS
- SYS:/APACHE2/APACHE2/CGI-BIN
- SYS:/APACHE2/MODULES nlm module
- HTTPD.CONF @@Value@@

- SEARCH ADD SYS:\APACHE2

SYS:/APACHE2

SYS

makefile "install"

DIST

NetWare

(

[Net](#)



```
load apache : . .
```

```
load address space = apache2 apache2
```

```
apache2 . NetWare
```

```
( Listen ) 80 .
```

```
error_log .
```

```
conf .
```

```
:
```

```
unload apache2
```

```
apache2 shutdown
```

```
unload :
```

```
unload address space = apache2 apache2
```

```
:
```

- -f

```
apache2 -f "vol:/my server/conf/my.conf"
```

```
apache -f test/test.conf
```

## ServerRoot .

-f , ( conf/  
SERVER\_CONFIG\_FILE .

:

- -C ServerRoot .
- -d .
- 
- server root.

server root sys:/apache2. -V

.

NetWare 2.0 .

. APACHE2 .

### **RESTART**

,

worker .

### **VERSION**

.

### **MODULES**

.

### **DIRECTIVES**

.

### **SETTINGS**

.

, .

### **SHUTDOWN**

.

### **HELP**

.

, .

"apache2 Help" .



conf . , NetWare

NetWare :

- NetWare ,  
: worker .

""\_ :

MaxRequestsPerChild - worker

MaxRequestsPerChild 0

NetWare 0

StartThreads -

StartThreads 50.

MinSpareThreads - (idle) worker

MinSpareThreads 10.

MaxSpareThreads - worker

MaxSpareThreads 100.

MaxThreads - worker

ThreadsPerChild 250.

ThreadStackSize - worker

ThreadStackSize 65536.

- NetWare .

- NetWare  
\\Apache2\modules

[LoadModule](#) . status :

```
LoadModule status_module modules/status.nlm
```

## NetWare :

- [CGIMapExtension](#) - CGI .
- [SecureListen](#) - SSL.
- [NWSSLTrustedCerts](#) -
- [NWSSLUpgradeable](#) - / SSL



```

MetroWerks CodeWarrior 6.x . Netware
sys:/Apache2 .

conf . conf HTTPD-S
HTTPD.CONF . HTTPD.CONF @@Value@@ .
conf/magic conf/mime.types . makefile
install .

```

:

NetWare 2.0 :

- Metrowerks CodeWarrior 6.0 [NetWare PDK 3.0](#) .
- [NetWare Libraries for C \(LibC\)](#)
- [LDAP Libraries for C](#)
- [ZLIB](#)
- AWK (awk, gawk) . AWK  
<http://developer.novell.com/ndk/apache.htm> .  
awk.exe .
- makefile  
<http://developer.novell.com/ndk/apache.htm> GNU make  
3.78.1 (GMake) .

## NetWare makefile :

- NOVELLLIBC

```
Set NOVELLLIBC=c:\novell\ndk\libc
```

NetWare Libraries for C SDK .

- METROWERKS

```
Set METROWERKS=C:\Program Files\Metrowerks\CodeWarrior
```

Metrowerks CodeWarrior .  
Files\Metrowerks\CodeWarrior, .

- LDAPSDK

```
Set LDAPSDK=c:\Novell\NDK\cldapsdk\NetWare\libc
```

LDAP Libraries for C .

- ZLIBSDK

```
Set ZLIBSDK=D:\NOVELL\zlib
```

ZLib .

- AP\_WORK \httpd-2.0 .
- APR\_WORK \httpd-2.0\srclib\apr
- AWK GNU make ( gmake.exe) PATH
- .
- \httpd-2.0\srclib\apr-util\uri "gmake -f nwgnumakefile" GENURI.nlm.
- GENURI.nlm NetWare SYS:

```
SYS:\genuri > sys:\uri_delims.h
```

- uri\_delims.h \httpd-2.0\srclib\apr-util\uri .
- \httpd-2.0\srclib\apr "gmake -f nwgnumakefile" APR .
- \httpd-2.0\srclib\pcre "gmake -f nwgnumakefile" DFTABLES.nlm.
- \httpd-2.0\server "gmake -f nwgnumakefile" GENCHARS.nlm.
- GENCHARS.nlm DFTABLES.nlm NetWare SYS:

```
SYS:\genchars > sys:\test_char.h
SYS:\dftables > sys:\chartables.c
```

- test\_char.h chartables.c \httpd-2.0\os\netware .
- \httpd-2.0 "gmake -f nwgnmakefile"

```
gmake -f nwgnmakefile install
```

```
install .
```

## make

- gmake -f nwgnmakefile \release .
- gmake -f nwgnmakefile DEBUG=1 \debug .
- gmake -f nwgnmakefile install \dist\Apache2 , , .
- gmake -f nwgnmakefile installdev install, \lib \include import .
- gmake -f nwgnmakefile clean DEBUG \release \debug .
- gmake -f nwgnmakefile clobber\_all clean .



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Platform Specific Notes](#)

# HPUX

Date: Wed, 05 Nov 1997 16:59:34 -0800  
From: Rick Jones <[raj@cup.hp.com](mailto:raj@cup.hp.com)>  
Reply-To: [raj@cup.hp.com](mailto:raj@cup.hp.com)  
Organization: Network Performance  
Subject: HP-UX tuning tips

HP-UX .

HP-UX 9.X: 10.20  
HP-UX 10.[00|01|10]: 10.20

HP-UX 10.20:

ARPA Transport . TCP .  
, 2 . adb \*disc\* .  
tcp\_hash\_size 32 disc 16 "  
"W" .

? <ftp://ftp.cup.hp.com/dist/networking/tools/connhist> ,  
TCP . (10 )  
SPECweb96 . <http://www.specbench.com>  
HP-UX 1000 SPECweb96 TIME\_WAIT 60  
60,000 TCP "" .

<ftp://ftp.cup.hp.com/dist/networking/misc/listenq>

PA-8000 , "chatr".  
<> ". GID MLOCK . MLOCK  
Setprivgrp(1m) . Glance

```
, mpctl()
.
FIN_WAIT_2 , nettune tcp_keepstar
. - 4 . tcp_hash_size ,
FIN_WAIT_2 ( 2) - .
, .
,
```

rick jones

<http://www.cup.hp.com/netperf/NetperfPage.html>



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Platform Specific Notes](#)

# EBCDIC

. .

2.0 . ,



### 1.3 EBCDIC

(-ASCII)

([BS2000/OSD](#) [SIEMENS](#) .  
POSIX ).

- [CERN-3.0](#) " "
- `prefork CERN` `accept-fork-ser`



---

EBCDIC (EBCDIC)  
CERN . HTML ( CERN  
(POSIX . grep sed POSIX  
) EBCDIC .  
" MIME " ( ).  
handler" .



---

BUFF

BUFF

:

- ( ASCII )

- content type / ( ASCII )

- ( ASCII )

- content type / ( )



1. #ifdef :

```
#ifdef CHARSET_EBCDIC
```

```
    EBCDIC . ,  
    HTTP .
```

```
#ifdef _OSD_POSIX
```

```
    SIEMENS BS2000/OSD . BS2000/
```

2. ASCII EBCDIC (BS2000 POSIX  
) HTTP

```
( GET , Header: , ) ASCII , (  
GIF , CGI . ) "" . ""  
"" , bgets() rvputs(), bg  
rvputs() .
```

```
( EBCDIC ASCII )
```

3. ( EBCDIC )

```
. ASCII escape \012 \015 :  
ASCII \n \r ASCII .  
; EBCDIC ASCII .
```

4. BUFF puts/write/get/gets

```
"ebcdic/ascii " , . (  
CGI) () :
```

```
EBCDIC CGI , ASCII  
(WWW : GIF). EBCDI
```

```
; type A  
EBCDIC .
```

5. (MIME type text/plain, text/html ) ASCII

, ( ASCII

.

:

.html ASCII text/html (
ASCII text/plain) :

```
AddType text/x-ascii-html .html
AddType text/x-ascii-plain .ascii
```

, text/foo MIME type AddType "text/x-ascii-foo" "ASCII" .

6. "" , GIF
. " rcp -b"

7. (, EBCDIC) ,

8. CGI CGI : Content-Typ
, GIF . wwwcount .



---

Content-Type: text/ .  
GIF , gzip .

PC ftp "binary" (  
( rcp -b ) rcp -b .

( , Content-Type: text/ )  
EBCDIC .

### **Server Side Include**

SSI EBCDIC . ASCII .



<u>core</u>	+	
<u>mod_access</u>	+	
<u>mod_actions</u>	+	
<u>mod_alias</u>	+	
<u>mod_asis</u>	+	
<u>mod_auth</u>	+	
<u>mod_auth_anon</u>	+	
<u>mod_auth_dbm</u>	?	libdb.a
<u>mod_autoindex</u>	+	
<u>mod_cern_meta</u>	?	
<u>mod_cgi</u>	+	
mod_digest	+	
<u>mod_dir</u>	+	
<u>mod_so</u>	-	
<u>mod_env</u>	+	
<u>mod_example</u>	-	( )
<u>mod_expires</u>	+	
<u>mod_headers</u>	+	
<u>mod_imagemap</u>	+	
<u>mod_include</u>	+	
<u>mod_info</u>	+	
mod_log_agent	+	
mod_log_config	+	
<u>mod_log_referer</u>	+	
<u>mod_mime</u>	+	
<u>mod_mime_magic</u>	?	

<u>mod_negotiation</u>	+	
<u>mod_proxy</u>	+	
<u>mod_rewrite</u>	+	
<u>mod_setenvif</u>	+	
<u>mod_speling</u>	+	
<u>mod_status</u>	+	
<u>mod_unique_id</u>	+	
<u>mod_userdir</u>	+	
<u>mod_usertrack</u>	?	



<a href="#">JK (mod_jserv)</a>	-	JAVA .
<a href="#">mod_php3</a>	+	mod_php3 LDAP, GD, FreeType .
<a href="#">mod_put</a>	?	
<a href="#">mod_session</a>	-	



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# httpd -

httpd

(HTTP) . (standalone)

httpd

[apachectl](#) ,

[2000, X](#)



[apachectl](#)



```
httpd [ -d serverroot ] [ -f config ] [ -C
directive ] [ -c directive ] [ -D parameter ] [ -
e level ] [ -E file ] [ -k
start|restart|graceful|stop ] [ -R directory ] [
-h ] [ -l ] [ -L ] [ -S ] [ -t ] [ -v ] [ -V ] [
-X ] [ -M ]
```

[Windows](#) :

```
httpd [ -k install|config|uninstall ] [ -n name ]
[ -w ]
```



```

-d serverroot
    ServerRoot serverroot . ServerRoot
    . /usr/local/apache2.

-f config
    config . config / ServerRoot
    . conf/httpd.conf.

-k start|restart|graceful|stop
    httpd , , .   .

-C directive
    directive .

-c directive
    directive .

-D parameter
    <IfDefine> parameter
    .

-e level
    LogLevel level .
    .

-E file
    file .

-R directory
    SHARED_CORE directory .

-h
    .

-l
    . LoadModule .

-L
    .

```

**-M**  
 .

**-S**  
 ( ).

**-t**  
 . ( ) 0 ( ) 0  
 . -D *DUMP\_VHOSTS* . -D  
*DUMP\_MODULES* .

**-v**  
 httpd .

**-V**  
 httpd .

**-X**  
 . , .

[Windows](#) :

**-k install|config|uninstall**  
 Windows NT ; ; .

**-n *name***  
*name*.

**-w**  
 .



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

ab -

ab (HTTP)

(benchmarking) .



[httpd](#)



```
ab [ -A auth-username:password ] [ -c concurrency ]  
 [ -C cookie-name=value ] [ -d ] [ -e csv-file ]  
 [ -g gnuplot-file ] [ -h ] [ -H custom-header ] [ -i ]  
 [ -k ] [ -n requests ] [ -p POST-file ] [ -P proxy-auth-username:password ] [ -q ] [ -s ] [ -S ]  
 [ -t timelimit ] [ -T content-type ] [ -v verbosity ] [ -V ] [ -w ] [ -x <table>-attributes ]  
 [ -X proxy[:port] ] [ -y <tr>-attributes ] [ -z <td>-attributes ] [http://]hostname[:port]/path
```



**-A *auth-username:password***  
 BASIC Authentication . : base64  
 ( , 401 ) .

**-c *concurrency***  
 . .

**-C *cookie-name=value***  
 Cookie: . *name=value* .

**-d**  
 "percentage served within XX [ms] table" . ( ).

**-e *csv-file***  
 ( ) (1% 100%) (CSV) .  
 " 'gnuplot' .

**-g *gnuplot-file***  
 'gnuplot' TSV (Tab separate values, )  
 . Gnuplot, IDL, Mathematica, Igor, Excel

**-h**  
 .

**-H *custom-header***  
 ( , "Accept  
 zip/zop;8bit") .

**-i**  
 GET HEAD .

**-k**  
 HTTP KeepAlive . , HTTP .  
 KeepAlive .

**-n *requests***  
 . .

**-p *POST-file***  
 POST .

**-P *proxy-auth-username:password***  
 BASIC Authentication . : base64  
 . ( , 401 ) .

**-q**  
 150 ab 10% 100 . -q  
 .

**-s**  
 ( ab -h ) http SSL https  
 . .

**-S**  
 , / .  
 ( ).

**-t *timelimit***  
 . -n 50000 .  
 .

**-T *content-type***  
 POST Content-type .

**-v *verbosity***  
 . 4 , 3 (404, 202, ) ,  
 (warning) (info) .

**-V**  
 .

**-w**  
 HTML .

**-x *<table>-attributes***  
 <table> . <table > .

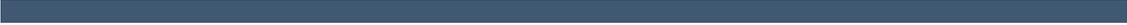
**-X *proxy[:port]***

.

**-y** *<tr>-attributes*  
    <tr> .

**-z** *<td>-attributes*  
    <td> .





· , ' ,  
·  
HTTP/1.x ; " .  
; , ab



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# apachectl -

```
apachectl (HTTP) .
apachectl . httpd
start, restart, stop httpd . apachectl
, httpd apachectl
apachectl 0, >0 .
```



[httpd](#)



---

, `apachectl httpd` .

**`apachectl`** [ *httpd-argument* ]

SysV init , `apachectl` .

**`apachectl`** *command*



---

SysV init- . [httpd](#) manpage .

**start**

httpd . . . apachectl -k start

**stop**

httpd . . . apachectl -k stop .

**restart**

httpd . , .  
configtest . . . apachectl -k restart

**fullstatus**

mod\_status . . . mod\_status  
, lynx . URL STATUSURL

**status**

. . . fullstatus , .

**graceful**

httpd (gracefully) . , .  
, , ,  
configtest . . . apachectl -k graceful

**configtest**

. . . Syntax Ok .  
apachectl -t .

**startssl**

apachectl -k start -DSSL .

SSL            httpd.conf   <IfDefine> .

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

|| [FAQ](#) ||



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# apxs - APache eXtenSion

apxs (HTTP) .  
, mod\_so LoadModule (DSO) .

DSO httpd  
apxs .

```
$ httpd -l
```

mod\_so . apxs DSO  
:

```
$ apxs -i -a -c mod_foo.c  
gcc -fpic -DSHARED_MODULE -I/path/to/apache/include -c mod_foo.c  
ld -Bshareable -o mod_foo.so mod_foo.o  
cp mod_foo.so /path/to/apache/modules/mod_foo.so  
chmod 755 /path/to/apache/modules/mod_foo.so  
[activating module `foo' in /path/to/apache/etc/httpd.conf]  
$ apachectl restart  
/path/to/apache/sbin/apachectl restart: httpd not running, trying  
to start  
[Tue Mar 31 11:27:55 1998] [debug] mod_so.c(303): loaded module  
foo_module  
/path/to/apache/sbin/apachectl restart: httpd started  
$ _
```

*files C (.c) (.o), (.a) .*  
C , .  
(PIC, position independent code) . GCC -fpic  
. C apxs .

DSO mod\_so  
src/modules/standard/mod\_so.c .



[apachectl](#)

[httpd](#)



---

**apxs -g** [ **-S** *name=value* ] **-n** *modname*

**apxs -q** [ **-S** *name=value* ] *query* ...

**apxs -c** [ **-S** *name=value* ] [ **-o** *dsofile* ] [ **-I** *incdir* ] [ **-D** *name=value* ] [ **-L** *libdir* ] [ **-l** *libname* ] [ **-Wc**,*compiler-flags* ] [ **-Wl**,*linker-flags* ] *files* ...

**apxs -i** [ **-S** *name=value* ] [ **-n** *modname* ] [ **-a** ] [ **-A** ] *dso-file* ...

**apxs -e** [ **-S** *name=value* ] [ **-n** *modname* ] [ **-a** ] [ **-A** ] *dso-file* ...



---

**-n *modname***

**-i** (install) **-g** (template generation) .  
.  
-g ,  
( ) .

**-q**

apxs . *query* : CC, CFLAGS,  
CFLAGS\_SHLIB, INCLUDEDIR, LD\_SHLIB,  
LDFLAGS\_SHLIB, LIBEXECDIR, LIBS\_SHLIB, SBINDIR,  
SYSCONFDIR, TARGET.

```
INC=-I`apxs -q INCLUDEDIR`
```

, C Makefile .

**-S *name=value***

apxs .

**(template)**

**-g**

*name* ( **-n** ) : *mod\_name*  
, apxs .  
Makefile.

**DSO**

**-C**  
files C (.c) (.o) , files  
dsofile . -o files  
mod\_name.so .

**-o dsofile**  
mod\_unknown.so . files

**-D name=value**  
define .

**-I incdir**  
include .

**-L libdir**

**-l libname**

**-Wc, compiler-flags**  
compiler-flags libtool --mode=compile

**-Wl, linker-flags**  
linker-flags libtool --mode=link .

## DSO

**-i**  
modules .

**-a**  
httpd.conf LoadModule

**-A**  
-a, LoadModule ( #) . ,

```
.  
-e . -a -A , -i  
httpd.conf .
```



```
mod_foo.c .
```

```
:
```

```
$ apxs -c mod_foo.c  
/path/to/libtool --mode=compile gcc ... -c mod_foo.c  
/path/to/libtool --mode=link gcc ... -o mod_foo.la mod_foo.slo  
$ _
```

```
LoadModule . apxs  
httpd.conf . :
```

```
$ apxs -i -a mod_foo.la  
/path/to/instldso.sh mod_foo.la /path/to/apache/modules  
/path/to/libtool --mode=install cp mod_foo.la  
/path/to/apache/modules ... chmod 755  
/path/to/apache/modules/mod_foo.so  
[/path/to/apache/conf/httpd.conf `foo' ]  
$ _
```

```
LoadModule foo_module modules/mod_foo.so
```

```
.
```

```
-A .
```

```
$ apxs -i -A mod_foo.c
```

```
apxs
```

```
Makefile :
```

```
$ apxs -g -n foo  
Creating [DIR] foo  
Creating [FILE] foo/Makefile  
Creating [FILE] foo/modules.mk  
Creating [FILE] foo/mod_foo.c  
Creating [FILE] foo/.deps  
$ _
```

```
:
```

```
$ cd foo
$ make all reload
apxs -c mod_foo.c
/path/to/libtool --mode=compile gcc ... -c mod_foo.c
/path/to/libtool --mode=link gcc ... -o mod_foo.la mod_foo.slo
apxs -i -a -n "foo" mod_foo.la
/path/to/instldso.sh mod_foo.la /path/to/apache/modules
/path/to/libtool --mode=install cp mod_foo.la
/path/to/apache/modules ... chmod 755
/path/to/apache/modules/mod_foo.so
[/path/to/apache/conf/httpd.conf `foo' ]
apachectl restart
/path/to/apache/sbin/apachectl restart: httpd not running,
trying to start
[Tue Mar 31 11:27:55 1998] [debug] mod_so.c(303): loaded module
foo_module
/path/to/apache/sbin/apachectl restart: httpd started
$ _
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

configure -

configure



---

configure .

**./configure** [*OPTION*]... [*VAR=VALUE*]...

(, CC, CFLAGS, ...), VAR=VALUE .



- 
- [...](#)
  - [...](#)
  - [...](#)
  - [...](#)
  - [...](#)

configure .

**-C**

**--config-cache**

--cache-file=config.cache .

**--cache-file=FILE**

FILE .

**-h**

**--help [short|recursive]**

. short . recursive

**-n**

**--no-create**

configure , . makefile

**-q**

**--quiet**

checking ... .

**--srcdir=DIR**

DIR . configure

**--silent**



```

--datadir=DIR
      DIR .      datadir      PREFIX/share.
      autoconf
.

--includedir=DIR
      C DIR .      includedir      EPREFIX/include.

--infodir=DIR
      info DIR .      infodir      PREFIX/info.
.

--libdir=DIR
      DIR .      libdir      EPREFIX/lib.

--libexecdir=DIR
      (,) DIR .      libexecdir      EPREFIX/libe:
.

--localstatedir=DIR
      DIR .      localstatedir      PREFIX/var.
      autoconf
.

--mandir=DIR
      man DIR .      mandir      EPREFIX/man.

--oldincludedir=DIR
      gcc C      DIR .      oldincludedir
      /usr/include. autoconf
.

--sbindir=DIR
      DIR .
      apachectl, suexec .      sbindir
      EPREFIX/sbin.

--sharedstatedir=DIR
      DIR .      sharedstatedir      PREFIX/com
      autoconf
.

--sysconfdir=DIR

```

```
    httpd.conf, mime.types          DIR .
sysconfdir      PREFIX/etc.
```

```
(cross-compile) .
```

```
--build=BUILD
```

```
config.guess .
```

```
--host=HOST
```

```
HOST BUILD.
```

```
--target=TARGET
```

```
TARGET . HOST.
```

```
--disable-FEATURE
```

```
FEATURE . --enable-FEATURE=no .
```

```
--enable-FEATURE[=ARG]
```

```
FEATURE . ARG yes.
```

```
--enable-MODULE=shared
```

```
DSO .
```

```
--enable-MODULE=static
```

```
configure foo --enable-foo
```

**--disable-actions**

mod\_actions

**--disable-alias**

mod\_alias

**--disable-asis**

mod\_asis as-is

**--disable-auth**

mod\_auth

HTTP Basic Authentication

**--disable-autoindex**

mod\_autoindex

**--disable-access**

mod\_access

**--disable-cgi**

MPM CGI

mod\_cgi

**--disable-cgid**

MPM worker perchild

mod\_cgid

CGI

**--disable-charset-lite**

mod\_charset\_lite

EBCDIC

**--disable-dir**

mod\_dir

**--disable-env**  
mod\_env / .

**--disable-http**  
 HTTP . http .  
 : .

**--disable-imagemap**  
mod\_imagemap imagemap .

**--disable-include**  
mod\_include Server Side Includes .

**--disable-log-config**  
mod\_log\_config .

**--disable-mime**  
mod\_mime (mime-type, , ,  
 . ( ) MIME .

**--disable-negotiation**  
mod\_negotiation .

**--disable-setenvif**  
mod\_setenvif .

**--disable-status**  
mod\_status / .

**--disable-userdir**  
mod\_userdir .

, most all

**--enable-auth-anon**

```

    mod_auth_anon
--enable-auth-dbm
    mod_auth_dbm DBM HTTP Basic
    Authentication .
--enable-auth-digest
    mod_auth_digest RFC2617 Digest authentication
.
--enable-authnz-ldap
    mod_authnz_ldap LDAP
.
--enable-cache
    mod_cache
. (storage management
module) (, mod_cache_disk mod mem cache)
.
--enable-cern-meta
    mod_cern_meta CERN
.
--enable-charset-lite
    mod_charset_lite EBCDIC
.
--enable-dav
    mod_dav WebDAV mod
. --enable-dav .
: mod_dav http .
--enable-dav-fs
    mod_dav_fs DAV
. --enable-dav .
--enable-deflate
    mod_deflate
.
--enable-disk-cache

```

```

    mod_cache_disk .
--enable-expires
    mod_expires Expires .
--enable-ext-filter
    mod_ext_filter .
--enable-file-cache
    mod_file_cache .
--enable-headers
    mod_headers HTTP .
--enable-info
    mod_info .
--enable-ldap
    mod_ldap LDAP .
--enable-logio
    mod_logio .
--enable-mem-cache
    mod_mem_cache .
--enable-mime-magic
    mod_mime_magic MIME type .
--enable-isapi
    mod_isapi isapi .
--enable-proxy
    mod_proxy / . CONNECT, FT
        mod_proxy_connect, mod_proxy_ftp,
    mod_proxy_http . --enable-proxy
    .
--enable-proxy-connect
    mod_proxy_connect CONNECT

```

```

    mod_proxy , --enable-proxy
--enable-proxy-ftp
    mod_proxy_ftp FTP .
    mod_proxy , --enable-proxy .
--enable-proxy-http
    mod_proxy_http HTTP .
    mod_proxy , --enable-proxy .
--enable-rewrite
    mod_rewrite URL .
--enable-so
    mod_so DSO . --enable-mods-shared
.
--enable-speling
    mod_spelling URL .
--enable-ssl
    mod_ssl SSL/TLS .
--enable-unique-id
    mod_unique_id .
--enable-usertrack
    mod_usertrack .
--enable-vhost-alias
    mod_vhost_alias .
, . .
.
--enable-bucketeer
    mod_bucketeer (bucket) .

```

```

--enable-case-filter
    mod_case_filter .

--enable-case-filter-in
    mod_case_filter_in .

--enable-echo
    mod_echo ECHO .

--enable-example
    mod_example .

--enable-optional-fn-export
    mod_optional_fn_export (exporter)
    .

--enable-optional-fn-import
    mod_optional_fn_import (importer)
    .

--enable-optional-hook-export
    mod_optional_hook_export (hook) .

--enable-optional-hook-import
    mod_optional_hook_import .

```

## MPM

```

:

--with-module=module-type:module-file
    .
    modules/module-type module-file
    .
    configure module-file
    .

```

DSO

[apxs](#) .

**--with-mpm=MPM**

.  
MPM [beos](#), [leader](#), [mpmt\\_os2](#), [perchild](#),  
[prefork](#), [threadpool](#), [worker](#) .

[MP](#)

**--enable-maintainer-mode**

**--enable-mods-shared=MODULE-LIST**

. , [LoadModule](#)  
  
*MODULE-LIST* .  
:

```
--enable-mods-shared='headers rewrite dav'
```

, all most . ,

```
--enable-mods-shared=most
```

DSO .

**--enable-modules=MODULE-LIST**

--enable-mods-shared , . ,  
httpd . [LoadModule](#) .

**--enable-v4-mapped**

IPv6 IPv4 .

**--with-port=PORT**

httpd . httpd.conf

**--with-program-name**

httpd.

**--with-PACKAGE[=ARG]**

PACKAGE . ARG yes.

**--without-PACKAGE**

PACKAGE . --with-PACKAGE=no .  
autoconf .

**--with-apr=DIR|FILE**

httpd Apache Portable Runtime (APR)  
 . APR configure  
config .APR , , .  
bin apr-config .

**--with-apr-util=DIR|FILE**

httpd Apache Portable Runtime Utilities (APU)  
 . APU  
config .APU , , .  
bin apu-config .

**--with-ssl=DIR**

mod\_ssl configure OpenSSL .  
SSL/TLS .

**--with-z=DIR**

(mod\_deflate ) conf:

`mod_authn_dbm mod_rewrite DBM RewriteMap`

`/ .APU SDBM`

`--with-gdbm[=path]`

`path , configure GNU`

`path configure path/lib path/include`

`path`

`--with-ndbm[=path]`

`--with-gdbm New DBM .`

`--with-berkeley-db[=path]`

`--with-gdbm Berkeley DB .`

```
DBM APU APU . --wit
APU DBM .
DBM . DBM .
```

`--enable-static-support`

`--enable-suexec`

`uid gid suexec .`

`suexec .`

`:`

`--enable-static-ab`

`ab .`

`--enable-static-checkgid`

```

    checkgid .
--enable-static-htdbm
    htdbm .
--enable-static-htdigest
    htdigest .
--enable-static-htpasswd
    htpasswd .
--enable-static-logresolve
    logresolve .
--enable-static-rotatelogs
    rotatelogs .

suexec
    suexec . suEXEC .
--with-suexec-bin
    suexec . --sbindir (  )
--with-suexec-caller
    suexec . httpd
--with-suexec-docroot
    suexec .
    datadir/htdocs.
--with-suexec-gidmin
    suexec GID . 100.
--with-suexec-logfile
    suexec . suexec_log,
--with-suexec-safepath
    suexec PATH .
    /usr/local/bin:/usr/bin:/bin.

```

```
--with-suexec-userdir
    suexec      ( ) .
    (mod_userdir) .

--with-suexec-uidmin
    suexec UID .          100.

--with-suexec-umask
    suexec          umask .
```



---

configure

**CC**

C .

**CFLAGS**

C .

**CPP**

C .

**CPPFLAGS**

C/C++ . ,

*Iincludedir* .

*includedir*

**LDFLAGS**

. ,

*libdir*

*-Llibc*



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# dbmmanage - DBM

dbmmanage HTTP basic authentication DBM  
dbmmanage

[htpasswd](#)

manpage [httpd](#)  
<http://httpd.apache.org/>



[httpd](#)

[mod\\_authn\\_dbm](#)

[mod\\_authz\\_dbm](#)



---

```
dbmmmanage [ encoding ] filename  
add|adduser|check|delete|update username [  
encpasswd [ group[,group...] [ comment ] ] ]
```

```
dbmmmanage filename view [ username ]
```

```
dbmmmanage filename import
```



---

**filename**

DBM . .db, .pag, .dir .

**username**

. username ( :) .

**encpasswd**

update add .  
, update ( .) .

**group**

. ( :) . ( .)  
, update ( .) .

**comment**

, . .

**-d**

crypt (Win32 Netware )

**-m**

MD5 (Win32 Netware )

**-s**

SHA1

**-p**

( )

**add**

encpasswd filename username .

**adduser**

filename username .

**check**

*filename username .*

**delete**

*filename username .*

**import**

STDIN *username:password* ( ) *filename*

.

**update**

*adduser , filename username .*

**view**

DBM . *username .*



DBM  
 SDBM, NDBM, GNU GDBM, Berkeley DB 2.  
*filename* dbmmanage  
 dbmmanage DBM  
 DBM , DBM .  
 dbmmanage @AnyDBM::ISA DBM .  
 Berkeley DB 2 dbmmanage  
 NDBM, GDBM, SDBM . dbmmanage  
 DBM . Perl dbmopen() Perl  
 @AnyDBM::ISA . DBM  
 . C  
 file DBM .



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## htcacheclean -

```
htcacheclean mod_cache_disk .  
(daemon) .  
. TERM INT .
```



mod\_cache\_disk



---

```
htcacheclean [ -D ] [ -v ] [ -r ] [ -n ] -ppath -  
llimit
```

```
htcacheclean -b [ -n ] [ -i ] -dinterval -ppath -  
llimit
```



---

**-dinterval**

*interval*

. -D, -v, -r .  
SIGTERM SIGINT .

**-D**

-d .

**-v**

. -d .

**-r**

( ).

**-n**

(nice) .  
(2)

. htcachec1

**-ppath**

*path* .

CacheRoot

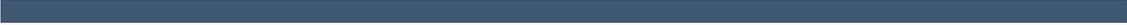
**-llimit**

*limit* . ( B )  
K, M .

**-i**

-d .





htcacheclean

(") 0 ,

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# htdigest - digest authentication

htdigest HTTP digest authentication , ,  
htdigest .

manpage . [httpd](http://httpd.apache.org/) digest authentication  
<http://httpd.apache.org/> .



[httpd](#)  
mod\_auth\_digest



---

```
htdigest [ -c ] passwdfile realm username
```



---

**-C**

*passwdfile* . *passwdfile* .

***passwdfile***

, , . -C ,

.

***realm***

.

***username***

*passwdfile* .

*username* .



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# htpasswd - basic authentication

htpasswd HTTP basic authentication

htpasswd ,

htpasswd

. DBM

htpasswd MD5

crypt() .

., MD5

manpage . [httpd](#)

<http://httpd.apache.org/>



[httpd](#)

SHA1 .



---

**htpasswd** [ **-c** ] [ **-m** ] [ **-D** ] *passwdfile username*

**htpasswd -b** [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-D** ]  
*passwdfile username password*

**htpasswd -n** [ **-m** | **-d** | **-s** | **-p** ] *username*

**htpasswd -nb** [ **-m** | **-d** | **-s** | **-p** ] *username*  
*password*



**-b**  
(batch) . , .  
.

**-c**  
*passwdfile* . *passwdfile* , . - n  
.

**-n**  
.  
*passwdfile* . -c .

**-m**  
MD5 . Windows, Netware, TPF .

**-d**  
crypt() . Windows, Netware, TPF  
. htpasswd , Window:  
TPF [httpd](#) .

**-s**  
SHA . LDAP (ldif) Netscape .

**-p**  
. htpasswd , Windows, Netware, TPF  
[httpd](#) .

**-D**  
. htpasswd .

***passwdfile***  
. -C , .

***username***  
*passwdfile* . *username* .  
.

***password***  
. -b .



---

htpasswd *passwdfile* ("")  
htpasswd 1, 2,  
3, 4, (, , , )  
5,  6,

.



```
htpasswd /usr/local/etc/apache/.htpasswd-users jsmith
```

```
jsmith . . Windows  
MD5 , crypt() .  
.
```

```
htpasswd -c /home/does/public_html/.htpasswd jane
```

```
jane . .  
htpasswd .
```

```
htpasswd -mb /usr/web/.htpasswd-all jones Pwd4Steve
```

```
( Pwd4Steve) MD5 .
```



---

htpasswd

URI

.,

-b .







| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

# logresolve - IP-



logresolve IP-

., IP

. IP ,



---

```
logresolve [ -s filename ] [ -c ] < access_log >  
access_log.new
```



---

**-s** *filename*

.

**-c**

logresolve DNS :IP

IP .



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## rotatelog -

```
rotatelog
```

```
CustomLog "|bin/rotatelog /var/logs/logfile 86400" common
```

```
/var/logs/logfile.nnnn . nnnn (
cron ). (24) .
```

```
CustomLog "|bin/rotatelog /var/logs/logfile 5M" common
```

```
5
```

```
ErrorLog "|bin/rotatelog /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 5M"
```

```
5
```

```
errorlog.YYYY-mm-dd-
```



---

```
rotatelogs [ -l ] logfile [ rotationtime [ offset  
] ] | [ filesizeM ]
```



---

**-1**

GMT . (BST DST ) GMT

-1

!

**logfile**

. logfile '%' strftime(3) . '%'  
. nnnnnnnnnn .

.

**rotationtime**

.

**offset**

UTC . 0 UTC . , UTC -5  
-300 .

**filesizeM**

M . rotationtime offset

.



strftime(3) .  
strftime(3) manpage .

%A	()
%a	() 3-
%B	()
%b	() 3-
%c	()
%d	2-
%H	2- (24 )
%I	2- (12 )
%j	3-
%M	2-
%m	2-
%p	() 12 am/pm
%S	2-
%U	2- ( )
%W	2- ( )
%w	1- ( )
%X	()
%x	()
%Y	4-
%y	2-

%Z	
%%	`%'



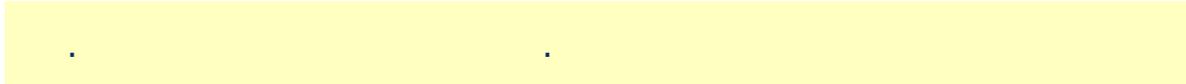
| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## Other Programs



manpage ,



perl cron .



## Optimizing the Perl log file

```
perl -e 'print "log\n";' >> log.log
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Miscellaneous Documentation](#)



.

.

.

:

- <http://purl.org/NET/http-errata> - HTTP/1.1
- <http://www.rfc-editor.org/errata.html> - RFC
- <http://ftp.ics.uci.edu/pub/ietf/http/#RFC> - HTTP RFC



.



---

IETF (recommendati

**RFC 1945 (Informational)**

(Hypertext Transfer Protocol, HTTP) , ,  
(application-level) .

**RFC 2616 (Standards Track)**

(Hypertext Transfer Protocol, HTTP) , ,  
. HTTP/1.1 .

**RFC 2396 (Standards Track)**

(Uniform Resource Identifier, URI)



(Hypertext Markup Language,  
IETF W3C :

HTML)

**RFC 2854 (Informational)**

HTML , W3C

"text/html" MIME

**HTML 4.01 (Errata)**

(Hypertext Marku

HTML 4 HTML 4.01 .

**HTML 3.2**

(Hypertext Markup Language,  
. HTML SGML .

HTML)

**XHTML 1.1 - XHTML** ()

Modularization of XHTML  
XHTML document type .

**XHTML 1.0 (Extensible HyperText Markup  
Language) (Second Edition)** ()

HTML 4 XML 1.0 XHTML 1.0  
DTD .

HTML 4



---

IETF :

[RFC 2617](#) (Draft standard)

Basic Access Authentication "HTTP/1.0".



ISO / :

**ISO 639-2**

ISO 639 . (639-1)  
( ) .

**ISO 3166-1**

ISO 3166-1 ISO 3166-1-alpha-2 ( ) .

**BCP 47 (Best Current Practice), RFC 3066**

**RFC 3282 (Standards Track)**

MIME RFC 822  
"Content-language:" , "Accept-Language:"



| | [FAQ](#) | |



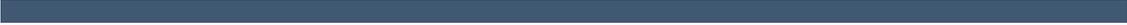
HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



.



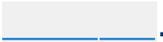


.



. . . . .

**MPM**

"MPM" 

MP

**Base**

"Base" ,

**Extension**

"Extension" .

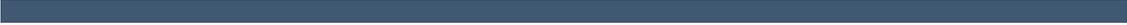
**Experimental**

"Experimental" ,

**External**

"External" (" ").

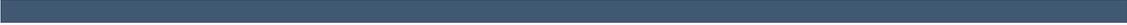




.

<IfModule> .

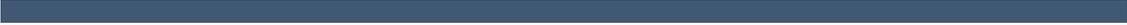




, [LoadModule](#) . |

.





2 ,

• ,

.



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



(Completion)

.



## **URL**

`http://www.example.com/path/to/file.html`  
(scheme), , Uniform Resource Local

## **URL-path**

`/path/to/file.html` *url* .

## **file-path**

`/usr/local/apache/htdocs/path/to/file.html`  
root . ,

## **directory-path**

`/usr/local/apache/htdocs/path/to/` root

## **filename**

`file.html` .

## **regex**

Perl (regular expression). *regex* .

## **extension**

*filename* .  
*filename* (extens  
, `file.html.en.html` .en .  
*extension* . , *extension*

## **MIME-type**

text/html major format

type minor format type

***env-variable***



(~~Content~~)

(  
 ,  
 .)  
 "None" .

httpd.conf





. . :

**(server config)**

( , httpd.conf) , <VirtualHost>  
<Directory> . .htaccess .

**(virtual host)**

<VirtualHost> .

**(directory)**

, <Directory>, <Location>  
<Files>, <Proxy> .

**.htaccess**

.htaccess .

.  
.  
, , .

(boolean) OR . ,

config, .htaccess" httpd.conf .htacc  
, <Directory> <VirtualHost> .

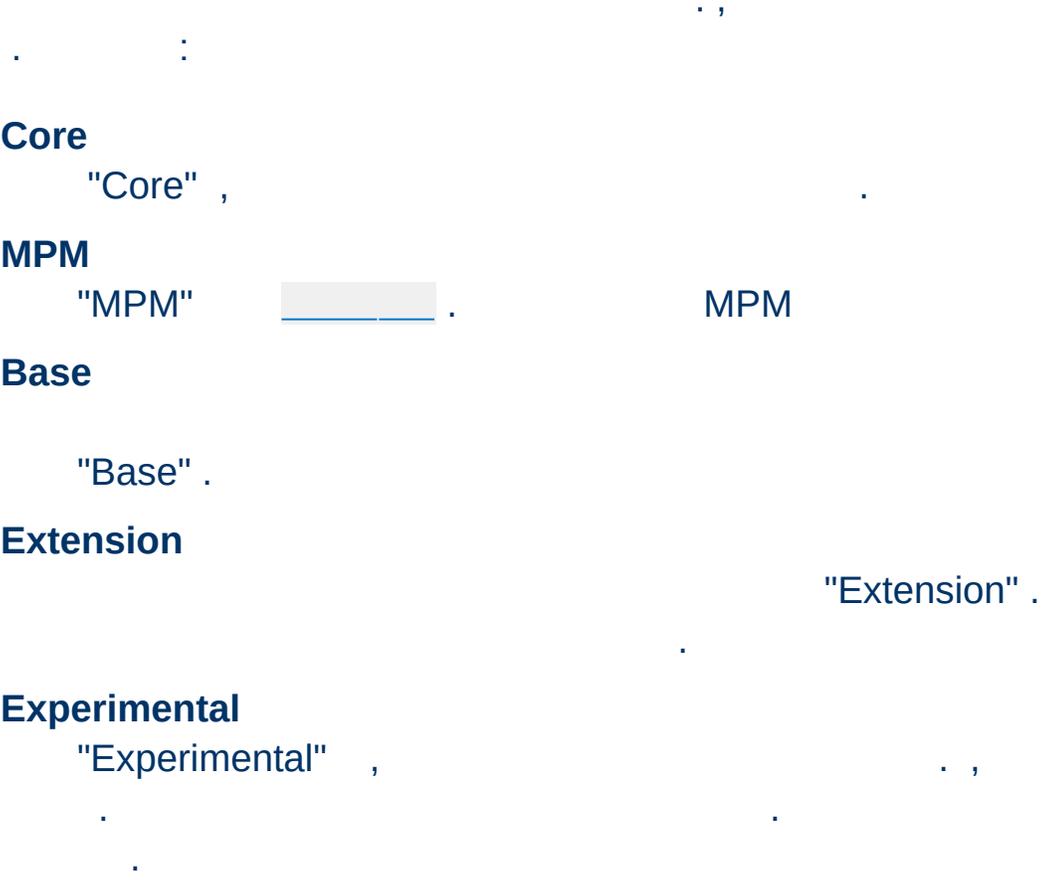


## Override (Override)

```
.htaccess          override .  
.htaccess          .
```

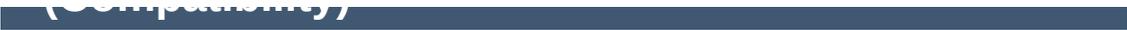
```
Overrides AllowOverride ,      ()  
         AllowOverride      .  
override .
```





.





2 ,

.. ,

.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Core Features

**Description:** Core Apache HTTP Server features that are always available

**Status:** Core



<b>Description:</b>	Configures optimizations for a Protocol's Listener Sockets
<b>Syntax:</b>	<code>AcceptFilter <i>protocol</i> <i>accept_filter</i></code>
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

This directive enables operating system specific optimizations for a listening socket by the `Protocol` type. The basic premise is for the kernel to not send a socket to the server process until either data is received or an entire HTTP Request is buffered. Only [FreeBSD's Accept Filters](#), Linux's more primitive `TCP_DEFER_ACCEPT`, and Windows' optimized `AcceptEx()` are currently supported.

Using `none` for an argument will disable any accept filters for that protocol. This is useful for protocols that require a server send data first, such as `ftp:` or `nntp:`

```
AcceptFilter nntp none
```

The default protocol names are `https` for port 443 and `http` for all other ports. To specify that another protocol is being used with a listening port, add the `protocol` argument to the [Listen](#) directive.

The default values on FreeBSD are:

```
AcceptFilter http httpready  
AcceptFilter https dataready
```

The `httpready` accept filter buffers entire HTTP requests at the

kernel level. Once an entire request is received, the kernel then sends it to the server. See the [accf\\_http\(9\)](#) man page for more details. Since HTTPS requests are encrypted, only the [accf\\_data\(9\)](#) filter is used.

The default values on Linux are:

```
AcceptFilter http data
AcceptFilter https data
```

Linux's TCP\_DEFER\_ACCEPT does not support buffering http requests. Any value besides none will enable TCP\_DEFER\_ACCEPT on that listener. For more details see the Linux [tcp\(7\)](#) man page.

The default values on Windows are:

```
AcceptFilter http connect
AcceptFilter https connect
```

Windows's mpm\_winnt interprets the AcceptFilter to toggle the AcceptEx() API, and does not support http protocol buffering. connect will use the AcceptEx() API, also retrieve the network endpoint addresses, but like none the connect option does not wait for the initial data transmission.

On Windows, none uses accept() rather than AcceptEx() and will not recycle sockets between connections. This is useful for network adapters with broken driver support, as well as some virtual network providers such as vpn drivers, or spam, virus or spyware filters.

**The data AcceptFilter (Windows)**

For versions 2.4.23 and prior, the Windows data accept filter waited until data had been transmitted and the initial data buffer and network endpoint addresses had been retrieved from the single AcceptEx() invocation. This implementation was subject to a denial of service attack and has been disabled.

Current releases of httpd default to the connect filter on Windows, and will fall back to connect if data is specified. Users of prior releases are encouraged to add an explicit setting of connect for their AcceptFilter, as shown above.

## See also

- [Protocol](#)



<b>Description:</b>	Resources accept trailing pathname information
<b>Syntax:</b>	AcceptPathInfo On Off Default
<b>Default:</b>	AcceptPathInfo Default
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

This directive controls whether requests that contain trailing pathname information that follows an actual filename (or non-existent file in an existing directory) will be accepted or rejected. The trailing pathname information can be made available to scripts in the `PATH_INFO` environment variable.

For example, assume the location `/test/` points to a directory that contains only the single file `here.html`. Then requests for `/test/here.html/more` and `/test/nothere.html/more` both collect `/more` as `PATH_INFO`.

The three possible arguments for the `AcceptPathInfo` directive are:

### **off**

A request will only be accepted if it maps to a literal path that exists. Therefore a request with trailing pathname information after the true filename such as `/test/here.html/more` in the above example will return a 404 NOT FOUND error.

### **On**

A request will be accepted if a leading path component maps to a file that exists. The above example `/test/here.html/more` will be accepted if `/test/here.html` maps to a valid file.

## Default

The treatment of requests with trailing pathname information is determined by the [handler](#) responsible for the request. The core handler for normal files defaults to rejecting PATH\_INFO requests. Handlers that serve scripts, such as [cgi-script](#) and [isapi-handler](#), generally accept PATH\_INFO by default.

The primary purpose of the AcceptPathInfo directive is to allow you to override the handler's choice of accepting or rejecting PATH\_INFO. This override is required, for example, when you use a [filter](#), such as [INCLUDES](#), to generate content based on PATH\_INFO. The core handler would usually reject the request, so you can use the following configuration to enable such a script:

```
<Files "mypaths.shtml">  
  Options +Includes  
  SetOutputFilter INCLUDES  
  AcceptPathInfo On  
</Files>
```



<b>Description:</b>	Name of the distributed configuration file
<b>Syntax:</b>	AccessFileName <i>filename</i> [ <i>filename</i> ] ...
<b>Default:</b>	AccessFileName .htaccess
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

While processing a request, the server looks for the first existing configuration file from this list of names in every directory of the path to the document, if distributed configuration files are [enabled for that directory](#). For example:

```
AccessFileName .acl
```

Before returning the document `/usr/local/web/index.html`, the server will read `/.acl`, `/usr/.acl`, `/usr/local/.acl` and `/usr/local/web/.acl` for directives unless they have been disabled with:

```
<Directory "/">
    AllowOverride None
</Directory>
```

## See also

- [AllowOverride](#)
- [Configuration Files](#)
- [.htaccess Files](#)



**Description:** Default charset parameter to be added when a response content-type is `text/plain` or `text/html`

**Syntax:** `AddDefaultCharset On|Off|charset`

**Default:** `AddDefaultCharset Off`

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** `FileInfo`

**Status:** Core

**Module:** core

This directive specifies a default value for the media type charset parameter (the name of a character encoding) to be added to a response if and only if the response's content-type is either `text/plain` or `text/html`. This should override any charset specified in the body of the response via a META element, though the exact behavior is often dependent on the user's client configuration. A setting of `AddDefaultCharset Off` disables this functionality. `AddDefaultCharset On` enables a default charset of `iso-8859-1`. Any other value is assumed to be the *charset* to be used, which should be one of the [IANA registered charset values](#) for use in Internet media types (MIME types). For example:

```
AddDefaultCharset utf-8
```

`AddDefaultCharset` should only be used when all of the text resources to which it applies are known to be in that character encoding and it is too inconvenient to label their charset individually. One such example is to add the charset parameter to resources containing generated content, such as legacy CGI scripts, that might be vulnerable to cross-site scripting attacks due to user-provided data being included in the output. Note, however,

that a better solution is to just fix (or delete) those scripts, since setting a default charset does not protect users that have enabled the "auto-detect character encoding" feature on their browser.

## See also

- [AddCharset](#)



<b>Description:</b>	Determines whether encoded path separators in URLs are allowed to be passed through
<b>Syntax:</b>	<code>AllowEncodedSlashes On   Off   NoDecode</code>
<b>Default:</b>	<code>AllowEncodedSlashes Off</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	NoDecode option available in 2.3.12 and later.

The `AllowEncodedSlashes` directive allows URLs which contain encoded path separators (`%2F` for `/` and additionally `%5C` for `\` on accordant systems) to be used in the path info.

With the default value, `Off`, such URLs are refused with a 404 (Not found) error.

With the value `On`, such URLs are accepted, and encoded slashes are decoded like all other encoded characters.

With the value `NoDecode`, such URLs are accepted, but encoded slashes are not decoded but left in their encoded state.

Turning `AllowEncodedSlashes On` is mostly useful when used in conjunction with `PATH_INFO`.

### Note

If encoded slashes are needed in path info, use of `NoDecode` is strongly recommended as a security measure. Allowing slashes to be decoded could potentially allow unsafe paths.

## See also

- AcceptPathInfo



<b>Description:</b>	Types of directives that are allowed in .htaccess files
<b>Syntax:</b>	AllowOverride All None  <i>directive-type</i> [ <i>directive-type</i> ] ...
<b>Default:</b>	AllowOverride None (2.3.9 and later), AllowOverride All (2.3.8 and earlier)
<b>Context:</b>	directory
<b>Status:</b>	Core
<b>Module:</b>	core

When the server finds an .htaccess file (as specified by [AccessFileName](#)), it needs to know which directives declared in that file can override earlier configuration directives.

#### Only available in <Directory> sections

[AllowOverride](#) is valid only in [<Directory>](#) sections specified without regular expressions, not in [<Location>](#), [<DirectoryMatch>](#) or [<Files>](#) sections.

When this directive is set to None and [AllowOverrideList](#) is set to None, [.htaccess](#) files are completely ignored. In this case, the server will not even attempt to read .htaccess files in the filesystem.

When this directive is set to All, then any directive which has the .htaccess [Context](#) is allowed in .htaccess files.

The *directive-type* can be one of the following groupings of directives. (See the [override class index](#) for an up-to-date listing of which directives are enabled by each *directive-type*.)

#### [AuthConfig](#)

Allow use of the authorization directives ([AuthDBMGroupFile](#), [AuthDBMUserFile](#), [AuthGroupFile](#), [AuthName](#), [AuthType](#), [AuthUserFile](#), [Require](#), etc.).

### **FileInfo**

Allow use of the directives controlling document types ([ErrorDocument](#), [ForceType](#), [LanguagePriority](#), [SetHandler](#), [SetInputFilter](#), [SetOutputFilter](#), and [mod\\_mime](#) Add\* and Remove\* directives), document meta data ([Header](#), [RequestHeader](#), [SetEnvIf](#), [SetEnvIfNoCase](#), [BrowserMatch](#), [CookieExpires](#), [CookieDomain](#), [CookieStyle](#), [CookieTracking](#), [CookieName](#)), [mod\\_rewrite](#) directives ([RewriteEngine](#), [RewriteOptions](#), [RewriteBase](#), [RewriteCond](#), [RewriteRule](#)), [mod\\_alias](#) directives ([Redirect](#), [RedirectTemp](#), [RedirectPermanent](#), [RedirectMatch](#)), and [Action](#) from [mod\\_actions](#).

### **Indexes**

Allow use of the directives controlling directory indexing ([AddDescription](#), [AddIcon](#), [AddIconByEncoding](#), [AddIconByType](#), [DefaultIcon](#), [DirectoryIndex](#), [FancyIndexing](#), [HeaderName](#), [IndexIgnore](#), [IndexOptions](#), [ReadmeName](#), etc.).

### **Limit**

Allow use of the directives controlling host access ([Allow](#), [Deny](#) and [Order](#)).

### **Nonfatal=[Override|Unknown|All]**

Allow use of AllowOverride option to treat syntax errors in .htaccess as nonfatal. Instead of causing an Internal Server Error, disallowed or unrecognised directives will be ignored and a warning logged:

- **Nonfatal=Override** treats directives forbidden by AllowOverride as nonfatal.
- **Nonfatal=Unknown** treats unknown directives as nonfatal. This covers typos and directives implemented by a module that's not present.
- **Nonfatal=All** treats both the above as nonfatal.

Note that a syntax error in a valid directive will still cause an internal server error.

### Security

Nonfatal errors may have security implications for .htaccess users. For example, if AllowOverride disallows AuthConfig, users' configuration designed to restrict access to a site will be disabled.

### Options[=*Option*,...]

Allow use of the directives controlling specific directory features ([Options](#) and [XBitHack](#)). An equal sign may be given followed by a comma-separated list, without spaces, of options that may be set using the [Options](#) command.

### Implicit disabling of Options

Even though the list of options that may be used in .htaccess files can be limited with this directive, as long as any [Options](#) directive is allowed any other inherited option can be disabled by using the non-relative syntax. In other words, this mechanism cannot force a specific option to remain set while allowing any others to be set.

```
AllowOverride Options=Indexes,MultiViews
```

Example:

```
AllowOverride AuthConfig Indexes
```

In the example above, all directives that are neither in the group `AuthConfig` nor `Indexes` cause an internal server error.

For security and performance reasons, do not set `AllowOverride` to anything other than `None` in your `<Directory "/">` block. Instead, find (or create) the `<Directory>` block that refers to the directory where you're actually planning to place a `.htaccess` file.

## See also

- [AccessFileName](#)
- [AllowOverrideList](#)
- [Configuration Files](#)
- [.htaccess Files](#)
- [Override Class Index for .htaccess](#)



<b>Description:</b>	Individual directives that are allowed in .htaccess files
<b>Syntax:</b>	AllowOverrideList None   <i>directive</i> [ <i>directive-type</i> ] ...
<b>Default:</b>	AllowOverrideList None
<b>Context:</b>	directory
<b>Status:</b>	Core
<b>Module:</b>	core

When the server finds an .htaccess file (as specified by [AccessFileName](#)), it needs to know which directives declared in that file can override earlier configuration directives.

#### Only available in <Directory> sections

[AllowOverrideList](#) is valid only in [<Directory>](#) sections specified without regular expressions, not in [<Location>](#), [<DirectoryMatch>](#) or [<Files>](#) sections.

When this directive is set to None and [AllowOverride](#) is set to None, then [.htaccess](#) files are completely ignored. In this case, the server will not even attempt to read .htaccess files in the filesystem.

Example:

```
AllowOverride None
AllowOverrideList Redirect RedirectMatch
```

In the example above, only the `Redirect` and `RedirectMatch` directives are allowed. All others will cause an internal server error.

Example:

```
AllowOverride AuthConfig  
AllowOverrideList CookieTracking CookieName
```

In the example above, [AllowOverride](#) grants permission to the AuthConfig directive grouping and [AllowOverrideList](#) grants permission to only two directives from the FileInfo directive grouping. All others will cause an internal server error.

## See also

- [AccessFileName](#)
- [AllowOverride](#)
- [Configuration Files](#)
- [.htaccess Files](#)



## CGIMapExtension Directive

<b>Description:</b>	Technique for locating the interpreter for CGI scripts
<b>Syntax:</b>	<code>CGIMapExtension <i>cgi-path</i> <i>.extension</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	NetWare only

This directive is used to control how Apache httpd finds the interpreter used to run CGI scripts. For example, setting `CGIMapExtension sys:\foo.nlm .foo` will cause all CGI script files with a `.foo` extension to be passed to the FOO interpreter.



<b>Description:</b>	Enables passing HTTP authorization headers to scripts as CGI variables
<b>Syntax:</b>	CGIPassAuth On Off
<b>Default:</b>	CGIPassAuth Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache HTTP Server 2.4.13 and later

**CGIPassAuth** allows scripts access to HTTP authorization headers such as `Authorization`, which is required for scripts that implement HTTP Basic authentication. Normally these HTTP headers are hidden from scripts. This is to disallow scripts from seeing user ids and passwords used to access the server when HTTP Basic authentication is enabled in the web server. This directive should be used when scripts are allowed to implement HTTP Basic authentication.

This directive can be used instead of the compile-time setting `SECURITY_HOLE_PASS_AUTHORIZATION` which has been available in previous versions of Apache HTTP Server.

The setting is respected by any modules which use `ap_add_common_vars()`, such as `mod_cgi`, `mod_cgid`, `mod_proxy_fcgi`, `mod_proxy_scgi`, and so on. Notably, it affects modules which don't handle the request in the usual sense but still use this API; examples of this are `mod_include` and `mod_ext_filter`. Third-party modules that don't use `ap_add_common_vars()` may choose to respect the setting as well.



<b>Description:</b>	Controls how some CGI variables are set
<b>Syntax:</b>	CGIVar <i>variable rule</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache HTTP Server 2.4.21 and later

This directive controls how some CGI variables are set.

**REQUEST\_URI** rules:

**original-uri (default)**

The value is taken from the original request line, and will not reflect internal redirects or subrequests which change the requested resource.

**current-uri**

The value reflects the resource currently being processed, which may be different than the original request from the client due to internal redirects or subrequests.



<b>Description:</b>	Enables the generation of Content -MD5 HTTP Response headers
<b>Syntax:</b>	ContentDigest On Off
<b>Default:</b>	ContentDigest Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Core
<b>Module:</b>	core

This directive enables the generation of Content -MD5 headers as defined in RFC1864 respectively RFC2616.

MD5 is an algorithm for computing a "message digest" (sometimes called "fingerprint") of arbitrary-length data, with a high degree of confidence that any alterations in the data will be reflected in alterations in the message digest.

The Content -MD5 header provides an end-to-end message integrity check (MIC) of the entity-body. A proxy or client may check this header for detecting accidental modification of the entity-body in transit. Example header:

```
Content-MD5: AuLb7Dp1rqRtRtxz2m9kRpA==
```

Note that this can cause performance problems on your server since the message digest is computed on every request (the values are not cached).

Content -MD5 is only sent for documents served by the [core](#), and not by any module. For example, SSI documents, output from CGI scripts, and byte range responses do not have this header.



<b>Description:</b>	Base directory for the server run-time files
<b>Syntax:</b>	<code>DefaultRuntimeDir</code> <i>directory-path</i>
<b>Default:</b>	<code>DefaultRuntimeDir</code> <code>DEFAULT_REL_RUNTIMEDIR</code> ( <code>logs/</code> )
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache 2.4.2 and later

The `DefaultRuntimeDir` directive sets the directory in which the server will create various run-time files (shared memory, locks, etc.). If set as a relative path, the full path will be relative to `ServerRoot`.

### Example

```
DefaultRuntimeDir scratch/
```

The default location of `DefaultRuntimeDir` may be modified by changing the `DEFAULT_REL_RUNTIMEDIR` #define at build time.

Note: `ServerRoot` should be specified before this directive is used. Otherwise, the default value of `ServerRoot` would be used to set the base directory.

### See also

- [the security tips](#) for information on how to properly set permissions on the `ServerRoot`



## DefaultType Directive

<b>Description:</b>	This directive has no effect other than to emit warnings if the value is not none. In prior versions, DefaultType would specify a default media type to assign to response content for which no other media type configuration could be found.
<b>Syntax:</b>	DefaultType <i>media-type none</i>
<b>Default:</b>	DefaultType none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	The argument none is available in Apache httpd 2.2.7 and later. All other choices are DISABLED for 2.3.x and later.

This directive has been disabled. For backwards compatibility of configuration files, it may be specified with the value none, meaning no default media type. For example:

```
DefaultType None
```

DefaultType None is only available in httpd-2.2.7 and later.

Use the mime.types configuration file and the [AddType](#) to configure media type assignments via file extensions, or the [ForceType](#) directive to configure the media type for specific resources. Otherwise, the server will send the response without a Content-Type header field and the recipient may attempt to guess the media type.



<b>Description:</b>	Define a variable
<b>Syntax:</b>	Define <i>parameter-name</i> [ <i>parameter-value</i> ]
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core

In its one parameter form, **Define** is equivalent to passing the `-D` argument to [httpd](#). It can be used to toggle the use of `<IfDefine>` sections without needing to alter `-D` arguments in any startup scripts.

In addition to that, if the second parameter is given, a config variable is set to this value. The variable can be used in the configuration using the `#{VAR}` syntax. The variable is always globally defined and not limited to the scope of the surrounding config section.

```
<IfDefine TEST>
  Define servername test.example.com
</IfDefine>
<IfDefine !TEST>
  Define servername www.example.com
  Define SSL
</IfDefine>

DocumentRoot "/var/www/#{servername}/htdocs"
```

Variable names may not contain colon ":" characters, to avoid clashes with [RewriteMap](#)'s syntax.

## Virtual Host scope and pitfalls

While this directive is supported in virtual host context, the changes it makes are visible to any later configuration directives, beyond any enclosing virtual host.

## See also

- [UnDefine](#)
- [IfDefine](#)



## Directory Directive

<b>Description:</b>	Enclose a group of directives that apply only to the named file-system directory, sub-directories, and their contents.
<b>Syntax:</b>	<code>&lt;Directory <i>directory-path</i>&gt; ...</code> <code>&lt;/Directory&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

`<Directory>` and `</Directory>` are used to enclose a group of directives that will apply only to the named directory, sub-directories of that directory, and the files within the respective directories. Any directive that is allowed in a directory context may be used. *Directory-path* is either the full path to a directory, or a wild-card string using Unix shell-style matching. In a wild-card string, `?` matches any single character, and `*` matches any sequences of characters. You may also use `[]` character ranges. None of the wildcards match a ``` character, so `<Directory */*/public_html">` will not match `/home/user/public_html`, but `<Directory "/home/*/public_html">` will match. Example:

```
<Directory "/usr/local/httpd/htdocs">  
    Options Indexes FollowSymLinks  
</Directory>
```

Directory paths *may* be quoted, if you like, however, it *must* be quoted if the path contains spaces. This is because a space would otherwise indicate the end of an argument.

Be careful with the *directory-path* arguments: They have to

literally match the filesystem path which Apache httpd uses to access the files. Directives applied to a particular `<Directory>` will not apply to files accessed from that same directory via a different path, such as via different symbolic links.

[Regular expressions](#) can also be used, with the addition of the `~` character. For example:

```
<Directory ~ "^/www/[0-9]{3}">
</Directory>
```

would match directories in `/www/` that consisted of three numbers.

If multiple (non-regular expression) `<Directory>` sections match the directory (or one of its parents) containing a document, then the directives are applied in the order of shortest match first, interspersed with the directives from the [.htaccess](#) files. For example, with

```
<Directory "/">
  AllowOverride None
</Directory>

<Directory "/home">
  AllowOverride FileInfo
</Directory>
```

for access to the document `/home/web/dir/doc.html` the steps are:

- Apply directive `AllowOverride None` (disabling `.htaccess` files).
- Apply directive `AllowOverride FileInfo` (for directory

/home).

- Apply any FileInfo directives in /home/.htaccess, /home/web/.htaccess and /home/web/dir/.htaccess in that order.

Regular expressions are not considered until after all of the normal sections have been applied. Then all of the regular expressions are tested in the order they appeared in the configuration file. For example, with

```
<Directory ~ "abc$">
  # ... directives here ...
</Directory>
```

the regular expression section won't be considered until after all normal `<Directory>`s and .htaccess files have been applied. Then the regular expression will match on /home/abc/public\_html/abc and the corresponding `<Directory>` will be applied.

**Note that the default access for `<Directory "/">` is to permit all access. This means that Apache httpd will serve any file mapped from an URL. It is recommended that you change this with a block such as**

```
<Directory "/">
  Require all denied
</Directory>
```

**and then override this for directories you *want* accessible. See the [Security Tips](#) page for more details.**

The directory sections occur in the httpd.conf file.

`<Directory>` directives cannot nest, and cannot appear in a

[<Limit>](#) or [<LimitExcept>](#) section.

## See also

- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received



## DirectoryMatch Directive

<b>Description:</b>	Enclose directives that apply to the contents of file-system directories matching a regular expression.
<b>Syntax:</b>	<code>&lt;DirectoryMatch regex&gt; ...</code> <code>&lt;/DirectoryMatch&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

`<DirectoryMatch>` and `</DirectoryMatch>` are used to enclose a group of directives which will apply only to the named directory (and the files within), the same as `<Directory>`. However, it takes as an argument a [regular expression](#). For example:

```
<DirectoryMatch "^/www/(.+)?[0-9]{3}/">
    # ...
</DirectoryMatch>
```

matches directories in `/www/` (or any subdirectory thereof) that consist of three numbers.

### Compatibility

Prior to 2.3.9, this directive implicitly applied to sub-directories (like `<Directory>`) and could not match the end of line symbol (`$`). In 2.3.9 and later, only directories that match the expression are affected by the enclosed directives.

### Trailing Slash

This directive applies to requests for directories that may or may not end in a trailing slash, so expressions that are anchored to the end of line (`$`) must be written with care.

From 2.4.8 onwards, named groups and backreferences are captured and written to the environment with the corresponding name prefixed with "MATCH\_" and in upper case. This allows elements of paths to be referenced from within [expressions](#) and modules like `mod_rewrite`. In order to prevent confusion, numbered (unnamed) backreferences are ignored. Use named groups instead.

```
<DirectoryMatch "^/var/www/combined/(?<site  
    Require ldap-group cn=%{env:MATCH_SITEN/  
</DirectoryMatch>
```

## See also

- [<Directory>](#) for a description of how regular expressions are mixed in with normal [<Directory>](#)s
- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received



<b>Description:</b>	Directory that forms the main document tree visible from the web
<b>Syntax:</b>	DocumentRoot <i>directory-path</i>
<b>Default:</b>	DocumentRoot "/usr/local/apache/htdocs"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

This directive sets the directory from which [httpd](#) will serve files. Unless matched by a directive like [Alias](#), the server appends the path from the requested URL to the document root to make the path to the document. Example:

```
DocumentRoot "/usr/web"
```

then an access to `http://my.example.com/index.html` refers to `/usr/web/index.html`. If the *directory-path* is not absolute then it is assumed to be relative to the [ServerRoot](#).

The [DocumentRoot](#) should be specified without a trailing slash.

## See also

- [Mapping URLs to Filesystem Locations](#)



<b>Description:</b>	Contains directives that apply only if the condition of a previous <code>&lt;If&gt;</code> or <code>&lt;ElseIf&gt;</code> section is not satisfied by a request at runtime
<b>Syntax:</b>	<code>&lt;Else&gt; . . . &lt;/Else&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Nested conditions are evaluated in 2.4.26 and later

The `<Else>` applies the enclosed directives if and only if the most recent `<If>` or `<ElseIf>` section in the same scope has not been applied. For example: In

```
<If "-z req('Host')">
  # ...
</If>
<Else>
  # ...
</Else>
```

The `<If>` would match HTTP/1.0 requests without a *Host:* header and the `<Else>` would match requests with a *Host:* header.

## See also

- `<If>`
- `<ElseIf>`
- [How <Directory>, <Location>, <Files> sections work](#) for an explanation of how these different sections are combined when a request is received. `<If>`, `<ElseIf>`, and `<Else>`

are applied last.



<b>Description:</b>	Contains directives that apply only if a condition is satisfied by a request at runtime while the condition of a previous <code>&lt;If&gt;</code> or <code>&lt;ElseIf&gt;</code> section is not satisfied
<b>Syntax:</b>	<code>&lt;ElseIf <i>expression</i>&gt; ... &lt;/ElseIf&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Nested conditions are evaluated in 2.4.26 and later

The `<ElseIf>` applies the enclosed directives if and only if both the given condition evaluates to true and the most recent `<If>` or `<ElseIf>` section in the same scope has not been applied. For example: In

```
<If "-R '10.1.0.0/16'">
  #...
</If>
<ElseIf "-R '10.0.0.0/8'">
  #...
</ElseIf>
<Else>
  #...
</Else>
```

The `<ElseIf>` would match if the remote address of a request belongs to the subnet 10.0.0.0/8 but not to the subnet 10.1.0.0/16.

## See also

- [Expressions in Apache HTTP Server](#), for a complete

reference and more examples.

- [<If>](#)
- [<Else>](#)
- [How <Directory>, <Location>, <Files> sections work](#) for an explanation of how these different sections are combined when a request is received. [<If>](#), [<ElseIf>](#), and [<Else>](#) are applied last.



<b>Description:</b>	Use memory-mapping to read files during delivery
<b>Syntax:</b>	EnableMMAP On Off
<b>Default:</b>	EnableMMAP On
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

This directive controls whether the [httpd](#) may use memory-mapping if it needs to read the contents of a file during delivery. By default, when the handling of a request requires access to the data within a file -- for example, when delivering a server-parsed file using [mod\\_include](#) -- Apache httpd memory-maps the file if the OS supports it.

This memory-mapping sometimes yields a performance improvement. But in some environments, it is better to disable the memory-mapping to prevent operational problems:

- On some multiprocessor systems, memory-mapping can reduce the performance of the [httpd](#).
- Deleting or truncating a file while [httpd](#) has it memory-mapped can cause [httpd](#) to crash with a segmentation fault.

For server configurations that are vulnerable to these problems, you should disable memory-mapping of delivered files by specifying:

```
EnableMMAP Off
```

For NFS mounted files, this feature may be disabled explicitly for the offending files by specifying:

---

```
<Directory "/path-to-nfs-files">  
  EnableMMAP Off  
</Directory>
```



<b>Description:</b>	Use the kernel sendfile support to deliver files to the client
<b>Syntax:</b>	EnableSendfile On Off
<b>Default:</b>	EnableSendfile Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Default changed to Off in version 2.3.9.

This directive controls whether [httpd](#) may use the sendfile support from the kernel to transmit file contents to the client. By default, when the handling of a request requires no access to the data within a file -- for example, when delivering a static file -- Apache httpd uses sendfile to deliver the file contents without ever reading the file if the OS supports it.

This sendfile mechanism avoids separate read and send operations, and buffer allocations. But on some platforms or within some filesystems, it is better to disable this feature to avoid operational problems:

- Some platforms may have broken sendfile support that the build system did not detect, especially if the binaries were built on another box and moved to such a machine with broken sendfile support.
- On Linux the use of sendfile triggers TCP-checksum offloading bugs on certain networking cards when using IPv6.
- On Linux on Itanium, `sendfile` may be unable to handle files over 2GB in size.
- With a network-mounted [DocumentRoot](#) (e.g., NFS, SMB, CIFS, FUSE), the kernel may be unable to serve the network

file through its own cache.

For server configurations that are not vulnerable to these problems, you may enable this feature by specifying:

```
EnableSendfile On
```

For network mounted files, this feature may be disabled explicitly for the offending files by specifying:

```
<Directory "/path-to-nfs-files">  
  EnableSendfile Off  
</Directory>
```

Please note that the per-directory and .htaccess configuration of `EnableSendfile` is not supported by `mod_cache_disk`. Only global definition of `EnableSendfile` is taken into account by the module.



<b>Description:</b>	Abort configuration parsing with a custom error message
<b>Syntax:</b>	Error <i>message</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	2.3.9 and later

If an error can be detected within the configuration, this directive can be used to generate a custom error message, and halt configuration parsing. The typical use is for reporting required modules which are missing from the configuration.

```
# Example
# ensure that mod_include is loaded
<IfModule !include_module>
    Error "mod_include is required by mod_foo"
</IfModule>

# ensure that exactly one of SSL,NOSSL is defined
<IfDefine SSL>
<IfDefine NOSSL>
    Error "Both SSL and NOSSL are defined. De"
</IfDefine>
</IfDefine>
<IfDefine !SSL>
<IfDefine !NOSSL>
    Error "Either SSL or NOSSL must be defined"
</IfDefine>
</IfDefine>
```



<b>Description:</b>	What the server will return to the client in case of an error
<b>Syntax:</b>	ErrorDocument <i>error-code document</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

In the event of a problem or error, Apache httpd can be configured to do one of four things,

1. output a simple hardcoded error message
2. output a customized message
3. internally redirect to a local *URL-path* to handle the problem/error
4. redirect to an external *URL* to handle the problem/error

The first option is the default, while options 2-4 are configured using the **ErrorDocument** directive, which is followed by the HTTP response code and a URL or a message. Apache httpd will sometimes offer additional information regarding the problem/error.

From 2.4.13, [expression syntax](#) can be used inside the directive to produce dynamic strings and URLs.

URLs can begin with a slash (/) for local web-paths (relative to the **DocumentRoot**), or be a full URL which the client can resolve.

Alternatively, a message can be provided to be displayed by the browser. Note that deciding whether the parameter is an URL, a path or a message is performed before any expression is parsed.

Examples:

```
ErrorDocument 500 http://example.com/cgi-bin/
ErrorDocument 404 /errors/bad_urls.php
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 "Sorry, can't allow you ac
ErrorDocument 403 Forbidden!
ErrorDocument 403 /errors/forbidden.py?refer
```

Additionally, the special value `default` can be used to specify Apache httpd's simple hardcoded message. While not required under normal circumstances, `default` will restore Apache httpd's simple hardcoded message for configurations that would otherwise inherit an existing `ErrorDocument`.

```
ErrorDocument 404 /cgi-bin/bad_urls.pl

<Directory "/web/docs">
    ErrorDocument 404 default
</Directory>
```

Note that when you specify an `ErrorDocument` that points to a remote URL (ie. anything with a method such as `http` in front of it), Apache HTTP Server will send a redirect to the client to tell it where to find the document, even if the document ends up being on the same server. This has several implications, the most important being that the client will not receive the original error status code, but instead will receive a redirect status code. This in turn can confuse web robots and other clients which try to determine if a URL is valid using the status code. In addition, if you use a remote URL in an `ErrorDocument 401`, the client will not know to prompt the user for a password since it will not receive the 401 status code. Therefore, **if you use an `ErrorDocument 401` directive, then it must refer to a local document.**

Microsoft Internet Explorer (MSIE) will by default ignore server-generated error messages when they are "too small" and substitute its own "friendly" error messages. The size threshold varies depending on the type of error, but in general, if you make your error document greater than 512 bytes, then MSIE will show the server-generated error rather than masking it. More information is available in Microsoft Knowledge Base article [Q294807](#).

Although most error messages can be overridden, there are certain circumstances where the internal messages are used regardless of the setting of [ErrorDocument](#). In particular, if a malformed request is detected, normal request processing will be immediately halted and the internal error message returned. This is necessary to guard against security problems caused by bad requests.

If you are using `mod_proxy`, you may wish to enable [ProxyErrorOverride](#) so that you can provide custom error messages on behalf of your Origin servers. If you don't enable `ProxyErrorOverride`, Apache `httpd` will not generate custom error documents for proxied content.

## See also

- [documentation of customizable responses](#)



## ErrorLog Directive

<b>Description:</b>	Location where the server will log errors
<b>Syntax:</b>	ErrorLog <i>file-path</i>  syslog[: [ <i>facility</i> ] [: <i>tag</i> ]]
<b>Default:</b>	ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows and OS/2)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The **ErrorLog** directive sets the name of the file to which the server will log any errors it encounters. If the *file-path* is not absolute then it is assumed to be relative to the **ServerRoot**.

```
ErrorLog "/var/log/httpd/error_log"
```

If the *file-path* begins with a pipe character "|" then it is assumed to be a command to spawn to handle the error log.

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

See the notes on [piped logs](#) for more information.

Using `syslog` instead of a filename enables logging via `syslogd(8)` if the system supports it. The default is to use `syslog` facility `local7`, but you can override this by using the `syslog:facility` syntax where *facility* can be one of the names usually documented in `syslog(1)`. The facility is effectively global, and if it is changed in individual virtual hosts, the final facility specified affects the entire server. Same rules apply for the `syslog` tag, which by default uses the Apache binary name, `httpd` in most cases. You can also override this by using the

`syslog::tag` syntax.

```
ErrorLog syslog:user  
ErrorLog syslog:user:httpd.srv1  
ErrorLog syslog::httpd.srv2
```

Additional modules can provide their own ErrorLog providers. The syntax is similar to the `syslog` example above.

**SECURITY:** See the [security tips](#) document for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server.

### Note

When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashes 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.

### See also

- [LogLevel](#)
- [Apache HTTP Server Log Files](#)



<b>Description:</b>	Format specification for error log entries
<b>Syntax:</b>	ErrorLogFormat [connection request] <i>format</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

**ErrorLogFormat** allows to specify what supplementary information is logged in the error log in addition to the actual log message.

```
#Simple example
ErrorLogFormat "[%t] [%l] [pid %P] %F: %E: |
```

Specifying `connection` or `request` as first parameter allows to specify additional formats, causing additional information to be logged when the first message is logged for a specific connection or request, respectively. This additional information is only logged once per connection/request. If a connection or request is processed without causing any log message, the additional information is not logged either.

It can happen that some format string items do not produce output. For example, the `Referer` header is only present if the log message is associated to a request and the log message happens at a time when the `Referer` header has already been read from the client. If no output is produced, the default behavior is to delete everything from the preceding space character to the next space character. This means the log line is implicitly divided into fields on non-whitespace to whitespace transitions. If a format string item does not produce output, the whole field is omitted. For example, if the remote address `%a` in the log format `[%t] [%l] [%a] %M`

is not available, the surrounding brackets are not logged either. Space characters can be escaped with a backslash to prevent them from delimiting a field. The combination '% ' (percent space) is a zero-width field delimiter that does not produce any output.

The above behavior can be changed by adding modifiers to the format string item. A - (minus) modifier causes a minus to be logged if the respective item does not produce any output. In once-per-connection/request formats, it is also possible to use the + (plus) modifier. If an item with the plus modifier does not produce any output, the whole line is omitted.

A number as modifier can be used to assign a log severity level to a format item. The item will only be logged if the severity of the log message is not higher than the specified log severity level. The number can range from 1 (alert) over 4 (warn) and 7 (debug) to 15 (trace8).

For example, here's what would happen if you added modifiers to the `%{Referer}i` token, which logs the Referer request header.

Modified Token	Meaning
<code>%- {Referer}i</code>	Logs a - if Referer is not set.
<code>%+{Referer}i</code>	Omits the entire line if Referer is not set.
<code>%4{Referer}i</code>	Logs the Referer only if the log message severity is higher than 4.

Some format string items accept additional parameters in braces.

Format String	Description
<code>%%</code>	The percent sign
<code>%a</code>	Client IP address and port of the request

<code>%{c}a</code>	Underlying peer IP address and port of the connection (see the <a href="#">mod_remoteip</a> module)
<code>%A</code>	Local IP-address and port
<code>%{name}e</code>	Request environment variable <i>name</i>
<code>%E</code>	APR/OS error status code and string
<code>%F</code>	Source file name and line number of the log call
<code>%{name}i</code>	Request header <i>name</i>
<code>%k</code>	Number of keep-alive requests on this connection
<code>%l</code>	Loglevel of the message
<code>%L</code>	Log ID of the request
<code>%{c}L</code>	Log ID of the connection
<code>%{C}L</code>	Log ID of the connection if used in connection scope, empty otherwise
<code>%m</code>	Name of the module logging the message
<code>%M</code>	The actual log message
<code>%{name}n</code>	Request note <i>name</i>
<code>%P</code>	Process ID of current process
<code>%T</code>	Thread ID of current thread
<code>%{g}T</code>	System unique thread ID of current thread (the same ID as displayed by e.g. <code>top</code> ; currently Linux only)
<code>%t</code>	The current time
<code>%{u}t</code>	The current time including micro-seconds
<code>%{cu}t</code>	The current time in compact ISO 8601 format, including micro-seconds
<code>%v</code>	The canonical <a href="#">ServerName</a> of the current server.
<code>%V</code>	The server name of the server serving the request according to the <a href="#">UseCanonicalName</a> setting.

\ (backslash space)	Non-field delimiting space
% (percent space)	Field delimiter (no output)

The log ID format %L produces a unique id for a connection or request. This can be used to correlate which log lines belong to the same connection or request, which request happens on which connection. A %L format string is also available in [mod\\_log\\_config](#) to allow to correlate access log entries with error log lines. If [mod\\_unique\\_id](#) is loaded, its unique id will be used as log ID for requests.

```
#Example (default format for threaded MPMs)
ErrorLogFormat "[%{u}t] [%-m:%l] [pid %P:tid
```

This would result in error messages such as:

```
[Thu May 12 08:28:57.652118 2011] [core:error] [pid 8777:tid
4326490112] [client ::1:58619] File does not exist:
/usr/local/apache2/htdocs/favicon.ico
```

Notice that, as discussed above, some fields are omitted entirely because they are not defined.

```
#Example (similar to the 2.2.x format)
ErrorLogFormat "[%t] [%l] %7F: %E: [client\
```

```
#Advanced example with request/connection log
ErrorLogFormat "[%{uc}t] [%-m:%-l] [R:%L] [C
ErrorLogFormat request "[%{uc}t] [R:%L] Reque
ErrorLogFormat request "[%{uc}t] [R:%L] UA:
ErrorLogFormat request "[%{uc}t] [R:%L] Refe
```

```
ErrorLogFormat connection "[%{uc}t] [C:%{c}I
```



## See also

- [ErrorLog](#)
- [LogLevel](#)
- [Apache HTTP Server Log Files](#)



<b>Description:</b>	Keep track of extended status information for each request
<b>Syntax:</b>	ExtendedStatus On Off
<b>Default:</b>	ExtendedStatus Off[*]
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

This option tracks additional data per worker about the currently executing request and creates a utilization summary. You can see these variables during runtime by configuring [mod\\_status](#). Note that other modules may rely on this scoreboard.

This setting applies to the entire server and cannot be enabled or disabled on a virtualhost-by-virtualhost basis. The collection of extended status information can slow down the server. Also note that this setting cannot be changed during a graceful restart.

Note that loading [mod\\_status](#) will change the default behavior to ExtendedStatus On, while other third party modules may do the same. Such modules rely on collecting detailed information about the state of all workers. The default is changed by [mod\\_status](#) beginning with version 2.3.6. The previous default was always Off.



## FileETag Directive

<b>Description:</b>	File attributes used to create the ETag HTTP response header for static files
<b>Syntax:</b>	FileETag <i>component</i> ...
<b>Default:</b>	FileETag MTime Size
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	The default used to be "INode MTime Size" in 2.3.14 and earlier.

The **FileETag** directive configures the file attributes that are used to create the ETag (entity tag) response header field when the document is based on a static file. (The ETag value is used in cache management to save network bandwidth.) The **FileETag** directive allows you to choose which of these -- if any -- should be used. The recognized keywords are:

### **INode**

The file's i-node number will be included in the calculation

### **MTime**

The date and time the file was last modified will be included

### **Size**

The number of bytes in the file will be included

### **All**

All available fields will be used. This is equivalent to:

```
FileETag INode MTime Size
```

### **None**

If a document is file-based, no ETag field will be included in

the response

The INode, MTime, and Size keywords may be prefixed with either + or -, which allow changes to be made to the default setting inherited from a broader scope. Any keyword appearing without such a prefix immediately and completely cancels the inherited setting.

If a directory's configuration includes FileETag INode MTime Size, and a subdirectory's includes FileETag -INode, the setting for that subdirectory (which will be inherited by any sub-subdirectories that don't override it) will be equivalent to FileETag MTime Size.

### Warning

Do not change the default for directories or locations that have WebDAV enabled and use `mod_dav_fs` as a storage provider. `mod_dav_fs` uses MTime Size as a fixed format for ETag comparisons on conditional requests. These conditional requests will break if the ETag format is changed via `FileETag`.

### Server Side Includes

An ETag is not generated for responses parsed by `mod_include` since the response entity can change without a change of the INode, MTime, or Size of the static file with embedded SSI directives.



<b>Description:</b>	Contains directives that apply to matched filenames
<b>Syntax:</b>	<code>&lt;Files filename&gt; ... &lt;/Files&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

The `<Files>` directive limits the scope of the enclosed directives by filename. It is comparable to the `<Directory>` and `<Location>` directives. It should be matched with a `</Files>` directive. The directives given within this section will be applied to any object with a basename (last component of filename) matching the specified filename. `<Files>` sections are processed in the order they appear in the configuration file, after the `<Directory>` sections and .htaccess files are read, but before `<Location>` sections. Note that `<Files>` can be nested inside `<Directory>` sections to restrict the portion of the filesystem they apply to.

The *filename* argument should include a filename, or a wild-card string, where ? matches any single character, and \* matches any sequences of characters.

```
<Files "cat.html">
    # Insert stuff that applies to cat.html
</Files>

<Files "?at.*">
    # This would apply to cat.html, bat.html
</Files>
```

[Regular expressions](#) can also be used, with the addition of the ~ character. For example:

```
<Files ~ "\.(gif|jpe?g|png)$">
    #...
</Files>
```

would match most common Internet graphics formats.

[<FilesMatch>](#) is preferred, however.

Note that unlike [<Directory>](#) and [<Location>](#) sections, [<Files>](#) sections can be used inside .htaccess files. This allows users to control access to their own files, at a file-by-file level.

## See also

- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received



<b>Description:</b>	Contains directives that apply to regular-expression matched filenames
<b>Syntax:</b>	<code>&lt;FilesMatch <i>regex</i>&gt; ... &lt;/FilesMatch&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

The `<FilesMatch>` directive limits the scope of the enclosed directives by filename, just as the `<Files>` directive does. However, it accepts a [regular expression](#). For example:

```
<FilesMatch ".+\.(gif|jpe?g|png)$">
    # ...
</FilesMatch>
```

would match most common Internet graphics formats.

The `.+` at the start of the regex ensures that files named `.png`, or `.gif`, for example, are not matched.

From 2.4.8 onwards, named groups and backreferences are captured and written to the environment with the corresponding name prefixed with "MATCH\_" and in upper case. This allows elements of files to be referenced from within [expressions](#) and modules like `mod_rewrite`. In order to prevent confusion, numbered (unnamed) backreferences are ignored. Use named groups instead.

```
<FilesMatch "^(?<sitename>[/]+)">
    require ldap-group cn=%{env:MATCH_SITENAME}
</FilesMatch>
```



## See also

- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received



## ForceType Directive

<b>Description:</b>	Forces all matching files to be served with the specified media type in the HTTP Content-Type header field
<b>Syntax:</b>	ForceType <i>media-type</i>   None
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

When placed into an `.htaccess` file or a `<Directory>`, or `<Location>` or `<Files>` section, this directive forces all matching files to be served with the content type identification given by *media-type*. For example, if you had a directory full of GIF files, but did not want to label them all with `.gif`, you might want to use:

```
ForceType image/gif
```

Note that this directive overrides other indirect media type associations defined in `mime.types` or via the `AddType`.

You can also override more general `ForceType` settings by using the value of `None`:

```
# force all files to be image/gif:
<Location "/images">
  ForceType image/gif
</Location>

# but normal mime-type associations here:
<Location "/images/mixed">
  ForceType None
</Location>
```

---

This directive primarily overrides the content types generated for static files served out of the filesystem. For resources other than static files, where the generator of the response typically specifies a Content-Type, this directive has no effect.

### Note

When explicit directives such as [SetHandler](#) or [AddHandler](#) do not apply to the current request, the internal handler name normally set by those directives is set to match the content type specified by this directive. This is a historical behavior that some third-party modules (such as mod\_php) may use "magic" content types used only to signal the module to take responsibility for the matching request. Configurations that rely on such "magic" types should be avoided by the use of [SetHandler](#) or [AddHandler](#).



<b>Description:</b>	Directory to write gmon.out profiling data to.
<b>Syntax:</b>	GprofDir /tmp/gprof/ /tmp/gprof/%
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

When the server has been compiled with gprof profiling support, **GprofDir** causes gmon.out files to be written to the specified directory when the process exits. If the argument ends with a percent symbol ("%"), subdirectories are created for each process id.

This directive currently only works with the [prefork](#) MPM.



<b>Description:</b>	Enables DNS lookups on client IP addresses
<b>Syntax:</b>	HostnameLookups On Off Double
<b>Default:</b>	HostnameLookups Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core

This directive enables DNS lookups so that host names can be logged (and passed to CGIs/SSIs in REMOTE\_HOST). The value Double refers to doing double-reverse DNS lookup. That is, after a reverse lookup is performed, a forward lookup is then performed on that result. At least one of the IP addresses in the forward lookup must match the original address. (In "tcpwrappers" terminology this is called PARANOID.)

Regardless of the setting, when [mod\\_authz\\_host](#) is used for controlling access by hostname, a double reverse lookup will be performed. This is necessary for security. Note that the result of this double-reverse isn't generally available unless you set HostnameLookups Double. For example, if only HostnameLookups On and a request is made to an object that is protected by hostname restrictions, regardless of whether the double-reverse fails or not, CGIs will still be passed the single-reverse result in REMOTE\_HOST.

The default is Off in order to save the network traffic for those sites that don't truly need the reverse lookups done. It is also better for the end users because they don't have to suffer the extra latency that a lookup entails. Heavily loaded sites should leave this directive Off, since DNS lookups can take considerable amounts of time. The utility [logresolve](#), compiled by default to the bin subdirectory of your installation directory, can be used to look up

host names from logged IP addresses offline.

Finally, if you have [hostname-based Require directives](#), a hostname lookup will be performed regardless of the setting of HostnameLookups.



<b>Description:</b>	Modify restrictions on HTTP Request Messages
<b>Syntax:</b>	HttpProtocolOptions [Strict Unsafe] [RegisteredMethods LenientMethods] [Allow0.9 Require1.0]
<b>Default:</b>	HttpProtocolOptions Strict LenientMethods Allow0.9
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	2.2.32 or 2.4.24 and later

This directive changes the rules applied to the HTTP Request Line ([RFC 7230 §3.1.1](#)) and the HTTP Request Header Fields ([RFC 7230 §3.2](#)), which are now applied by default or using the `Strict` option. Due to legacy modules, applications or custom user-agents which must be deprecated the `Unsafe` option has been added to revert to the legacy behaviors.

These rules are applied prior to request processing, so must be configured at the global or default (first) matching virtual host section, by IP/port interface (and not by name) to be honored.

The directive accepts three parameters from the following list of choices, applying the default to the ones not specified:

### **Strict|Unsafe**

Prior to the introduction of this directive, the Apache HTTP Server request message parsers were tolerant of a number of forms of input which did not conform to the protocol. [RFC 7230 §9.4 Request Splitting](#) and [§9.5 Response Smuggling](#) call out only two of the potential risks of accepting non-conformant request messages, while [RFC 7230 §3.5](#) "Message Parsing Robustness" identify the risks of accepting

obscure whitespace and request message formatting. As of the introduction of this directive, all grammar rules of the specification are enforced in the default `Strict` operating mode, and the strict whitespace suggested by section 3.5 is enforced and cannot be relaxed.

### Security risks of Unsafe

Users are strongly cautioned against toggling the `Unsafe` mode of operation, particularly on outward-facing, publicly accessible server deployments. If an interface is required for faulty monitoring or other custom service consumers running on an intranet, users should toggle the `Unsafe` option only on a specific virtual host configured to service their internal private network.

### Example of a request leading to HTTP 400 with Strict mode

```
# Missing CRLF
GET / HTTP/1.0\n\n
```

### Command line tools and CRLF

Some tools need to be forced to use CRLF, otherwise `httpd` will return a HTTP 400 response like described in the above use case. For example, the **OpenSSL `s_client` needs the `-crlf` parameter to work properly.**

The [DumpIOInput](#) directive can help while reviewing the HTTP request to identify issues like the absence of CRLF.

### RegisteredMethods|LenientMethods

[RFC 7231 §4.1](#) "Request Methods" "Overview" requires that origin servers shall respond with a HTTP 501 status code

when an unsupported method is encountered in the request line. This already happens when the `LenientMethods` option is used, but administrators may wish to toggle the `RegisteredMethods` option and register any non-standard methods using the `RegisterHttpMethod` directive, particularly if the `Unsafe` option has been toggled.

### Forward Proxy compatibility

The `RegisteredMethods` option should **not** be toggled for forward proxy hosts, as the methods supported by the origin servers are unknown to the proxy server.

### Example of a request leading to HTTP 501 with `LenientMethods` mode

```
# Unknown HTTP method
wOw / HTTP/1.0\r\n\r\n

# Lowercase HTTP method
get / HTTP/1.0\r\n\r\n
```

### `Allow0.9|Require1.0`

[RFC 2616 §19.6](#) "Compatibility With Previous Versions" had encouraged HTTP servers to support legacy HTTP/0.9 requests. RFC 7230 supersedes this with "The expectation to support HTTP/0.9 requests has been removed" and offers additional comments in [RFC 7230 Appendix A](#). The `Require1.0` option allows the user to remove support of the default `Allow0.9` option's behavior.

### Example of a request leading to HTTP 400 with `Require1.0` mode

```
# Unsupported HTTP version
GET /\r\n\r\n
```

Reviewing the messages logged to the [ErrorLog](#), configured with [LogLevel](#) debug level, can help identify such faulty requests along with their origin. Users should pay particular attention to the 400 responses in the access log for invalid requests which were unexpectedly rejected.



<b>Description:</b>	Contains directives that apply only if a condition is satisfied by a request at runtime
<b>Syntax:</b>	<If <i>expression</i> > ... </If>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Nested conditions are evaluated in 2.4.26 and later

The <If> directive evaluates an expression at runtime, and applies the enclosed directives if and only if the expression evaluates to true. For example:

```
<If "-z req('Host')">
```

would match HTTP/1.0 requests without a *Host:* header. Expressions may contain various shell-like operators for string comparison (==, !=, <, ...), integer comparison (-eq, -ne, ...), and others (-n, -z, -f, ...). It is also possible to use regular expressions,

```
<If "%{QUERY_STRING} =~ /(delete|commit)=.*">
```

shell-like pattern matches and many other operations. These operations can be done on request headers (req), environment variables (env), and a large number of other properties. The full documentation is available in [Expressions in Apache HTTP Server](#).

Only directives that support the [directory context](#) can be used

within this configuration section.

Certain variables, such as `CONTENT_TYPE` and other response headers, are set after `<If>` conditions have already been evaluated, and so will not be available to use in this directive.

## See also

- [Expressions in Apache HTTP Server](#), for a complete reference and more examples.
- [<ElseIf>](#)
- [<Else>](#)
- [How <Directory>, <Location>, <Files> sections work](#) for an explanation of how these different sections are combined when a request is received. `<If>`, `<ElseIf>`, and `<Else>` are applied last.



<b>Description:</b>	Encloses directives that will be processed only if a test is true at startup
<b>Syntax:</b>	<code>&lt;IfDefine [!]parameter-name&gt; ... &lt;/IfDefine&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

The `<IfDefine test>...</IfDefine>` section is used to mark directives that are conditional. The directives within an `<IfDefine>` section are only processed if the *test* is true. If *test* is false, everything between the start and end markers is ignored.

The *test* in the `<IfDefine>` section directive can be one of two forms:

- *parameter-name*
- `!parameter-name`

In the former case, the directives between the start and end markers are only processed if the parameter named *parameter-name* is defined. The second format reverses the test, and only processes the directives if *parameter-name* is **not** defined.

The *parameter-name* argument is a define as given on the [httpd](#) command line via `-Dparameter` at the time the server was started or by the `Define` directive.

`<IfDefine>` sections are nest-able, which can be used to implement simple multiple-parameter tests. Example:

```
httpd -DReverseProxy -DUseCache -DMemCache ...
```

```
<IfDefine ReverseProxy>
  LoadModule proxy_module    modules/mod_proxy
  LoadModule proxy_http_module  modules/mod_proxy_http
  <IfDefine UseCache>
    LoadModule cache_module    modules/mod_cache
    <IfDefine MemCache>
      LoadModule mem_cache_module  modules/mod_mem_cache
    </IfDefine>
    <IfDefine !MemCache>
      LoadModule cache_disk_module  modules/mod_cache_disk
    </IfDefine>
  </IfDefine>
</IfDefine>
```



<b>Description:</b>	Encloses directives that are processed conditional on the presence or absence of a specific module
<b>Syntax:</b>	<code>&lt;IfModule [!]module-file module-identifier&gt; ... &lt;/IfModule&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Module identifiers are available in version 2.1 and later.

The `<IfModule test>...</IfModule>` section is used to mark directives that are conditional on the presence of a specific module. The directives within an `<IfModule>` section are only processed if the *test* is true. If *test* is false, everything between the start and end markers is ignored.

The *test* in the `<IfModule>` section directive can be one of two forms:

- *module*
- *!module*

In the former case, the directives between the start and end markers are only processed if the module named *module* is included in Apache httpd -- either compiled in or dynamically loaded using [LoadModule](#). The second format reverses the test, and only processes the directives if *module* is **not** included.

The *module* argument can be either the module identifier or the file name of the module, at the time it was compiled. For example, `rewrite_module` is the identifier and `mod_rewrite.c` is the file

name. If a module consists of several source files, use the name of the file containing the string `STANDARD20_MODULE_STUFF`.

`<IfModule>` sections are nest-able, which can be used to implement simple multiple-module tests.

This section should only be used if you need to have one configuration file that works whether or not a specific module is available. In normal operation, directives need not be placed in `<IfModule>` sections.



<b>Description:</b>	Includes other configuration files from within the server configuration files
<b>Syntax:</b>	Include <i>file-path directory-path wildcard</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Directory wildcard matching available in 2.3.6 and later

This directive allows inclusion of other configuration files from within the server configuration files.

Shell-style (`fnmatch()`) wildcard characters can be used in the filename or directory parts of the path to include several files at once, in alphabetical order. In addition, if `Include` points to a directory, rather than a file, Apache httpd will read all files in that directory and any subdirectory. However, including entire directories is not recommended, because it is easy to accidentally leave temporary files in a directory that can cause `httpd` to fail. Instead, we encourage you to use the wildcard syntax shown below, to include files that match a particular pattern, such as `*.conf`, for example.

The `Include` directive will **fail with an error** if a wildcard expression does not match any file. The `IncludeOptional` directive can be used if non-matching wildcards should be ignored.

The file path specified may be an absolute path, or may be relative to the `ServerRoot` directory.

Examples:

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

Or, providing paths relative to your [ServerRoot](#) directory:

```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

Wildcards may be included in the directory or file portion of the path. This example will fail if there is no subdirectory in `conf/vhosts` that contains at least one `*.conf` file:

```
Include conf/vhosts/*/*.conf
```

Alternatively, the following command will just be ignored in case of missing files or directories:

```
IncludeOptional conf/vhosts/*/*.conf
```

## See also

- [IncludeOptional](#)
- [apachectl](#)



## IncludeOptional Directive

<b>Description:</b>	Includes other configuration files from within the server configuration files
<b>Syntax:</b>	<code>IncludeOptional file-path directory-path wildcard</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in 2.3.6 and later

This directive allows inclusion of other configuration files from within the server configuration files. It works identically to the [Include](#) directive, with the exception that if wildcards do not match any file or directory, the [IncludeOptional](#) directive will be silently ignored instead of causing an error.

### See also

- [Include](#)
- [apachectl](#)



<b>Description:</b>	Enables HTTP persistent connections
<b>Syntax:</b>	KeepAlive On Off
<b>Default:</b>	KeepAlive On
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The Keep-Alive extension to HTTP/1.0 and the persistent connection feature of HTTP/1.1 provide long-lived HTTP sessions which allow multiple requests to be sent over the same TCP connection. In some cases this has been shown to result in an almost 50% speedup in latency times for HTML documents with many images. To enable Keep-Alive connections, set `KeepAlive On`.

For HTTP/1.0 clients, Keep-Alive connections will only be used if they are specifically requested by a client. In addition, a Keep-Alive connection with an HTTP/1.0 client can only be used when the length of the content is known in advance. This implies that dynamic content such as CGI output, SSI pages, and server-generated directory listings will generally not use Keep-Alive connections to HTTP/1.0 clients. For HTTP/1.1 clients, persistent connections are the default unless otherwise specified. If the client requests it, chunked encoding will be used in order to send content of unknown length over persistent connections.

When a client uses a Keep-Alive connection, it will be counted as a single "request" for the [MaxConnectionsPerChild](#) directive, regardless of how many requests are sent using the connection.

## See also

- [MaxKeepAliveRequests](#)



<b>Description:</b>	Amount of time the server will wait for subsequent requests on a persistent connection
<b>Syntax:</b>	KeepAliveTimeout <i>num</i> [ms]
<b>Default:</b>	KeepAliveTimeout 5
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The number of seconds Apache httpd will wait for a subsequent request before closing the connection. By adding a postfix of ms the timeout can be also set in milliseconds. Once a request has been received, the timeout value specified by the [Timeout](#) directive applies.

Setting [KeepAliveTimeout](#) to a high value may cause performance problems in heavily loaded servers. The higher the timeout, the more server processes will be kept occupied waiting on connections with idle clients.

If [KeepAliveTimeout](#) is **not** set for a name-based virtual host, the value of the first defined virtual host best matching the local IP and port will be used.



**Description:** Restrict enclosed access controls to only certain HTTP methods

**Syntax:** `<Limit method [method] ... > ...`  
`</Limit>`

**Context:** directory, .htaccess

**Override:** AuthConfig, Limit

**Status:** Core

**Module:** core

Access controls are normally effective for **all** access methods, and this is the usual desired behavior. **In the general case, access control directives should not be placed within a `<Limit>` section.**

The purpose of the `<Limit>` directive is to restrict the effect of the access controls to the nominated HTTP methods. For all other methods, the access restrictions that are enclosed in the `<Limit>` bracket **will have no effect**. The following example applies the access control only to the methods POST, PUT, and DELETE, leaving all other methods unprotected:

```
<Limit POST PUT DELETE>
  Require valid-user
</Limit>
```

The method names listed can be one or more of: GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, and UNLOCK. **The method name is case-sensitive**. If GET is used, it will also restrict HEAD requests. The TRACE method cannot be limited (see [TraceEnable](#)).

A `<LimitExcept>` section should always be used in

preference to a `<Limit>` section when restricting access, since a `<LimitExcept>` section provides protection against arbitrary methods.

The `<Limit>` and `<LimitExcept>` directives may be nested. In this case, each successive level of `<Limit>` or `<LimitExcept>` directives must further restrict the set of methods to which access controls apply.

When using `<Limit>` or `<LimitExcept>` directives with the `Require` directive, note that the first `Require` to succeed authorizes the request, regardless of the presence of other `Require` directives.

For example, given the following configuration, all users will be authorized for POST requests, and the `Require group editors` directive will be ignored in all cases:

```
<LimitExcept GET>
  Require valid-user
</LimitExcept>
<Limit POST>
  Require group editors
</Limit>
```



<b>Description:</b>	Restrict access controls to all HTTP methods except the named ones
<b>Syntax:</b>	<code>&lt;LimitExcept <i>method</i> [<i>method</i>] ... &gt;</code> <code>... &lt;/LimitExcept&gt;</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig, Limit
<b>Status:</b>	Core
<b>Module:</b>	core

`<LimitExcept>` and `</LimitExcept>` are used to enclose a group of access control directives which will then apply to any HTTP access method **not** listed in the arguments; i.e., it is the opposite of a `<Limit>` section and can be used to control both standard and nonstandard/unrecognized methods. See the documentation for `<Limit>` for more details.

For example:

```
<LimitExcept POST GET>
  Require valid-user
</LimitExcept>
```



<b>Description:</b>	Determine maximum number of internal redirects and nested subrequests
<b>Syntax:</b>	<code>LimitInternalRecursion <i>number</i> [<i>number</i>]</code>
<b>Default:</b>	<code>LimitInternalRecursion 10</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

An internal redirect happens, for example, when using the [Action](#) directive, which internally redirects the original request to a CGI script. A subrequest is Apache httpd's mechanism to find out what would happen for some URI if it were requested. For example, [mod\\_dir](#) uses subrequests to look for the files listed in the [DirectoryIndex](#) directive.

[LimitInternalRecursion](#) prevents the server from crashing when entering an infinite loop of internal redirects or subrequests. Such loops are usually caused by misconfigurations.

The directive stores two different limits, which are evaluated on per-request basis. The first *number* is the maximum number of internal redirects that may follow each other. The second *number* determines how deeply subrequests may be nested. If you specify only one *number*, it will be assigned to both limits.

```
LimitInternalRecursion 5
```



<b>Description:</b>	Restricts the total size of the HTTP request body sent from the client
<b>Syntax:</b>	<code>LimitRequestBody</code> <i>bytes</i>
<b>Default:</b>	<code>LimitRequestBody</code> 0
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

This directive specifies the number of *bytes* from 0 (meaning unlimited) to 2147483647 (2GB) that are allowed in a request body. See the note below for the limited applicability to proxy requests.

The `LimitRequestBody` directive allows the user to set a limit on the allowed size of an HTTP request message body within the context in which the directive is given (server, per-directory, per-file or per-location). If the client request exceeds that limit, the server will return an error response instead of servicing the request. The size of a normal request message body will vary greatly depending on the nature of the resource and the methods allowed on that resource. CGI scripts typically use the message body for retrieving form information. Implementations of the PUT method will require a value at least as large as any representation that the server wishes to accept for that resource.

This directive gives the server administrator greater control over abnormal client request behavior, which may be useful for avoiding some forms of denial-of-service attacks.

If, for example, you are permitting file upload to a particular location and wish to limit the size of the uploaded file to 100K, you might use the following directive:

LimitRequestBody 102400

For a full description of how this directive is interpreted by proxy requests, see the [mod\\_proxy](#) documentation.



<b>Description:</b>	Limits the number of HTTP request header fields that will be accepted from the client
<b>Syntax:</b>	<code>LimitRequestFields</code> <i>number</i>
<b>Default:</b>	<code>LimitRequestFields</code> 100
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

*Number* is an integer from 0 (meaning unlimited) to 32767. The default value is defined by the compile-time constant `DEFAULT_LIMIT_REQUEST_FIELDS` (100 as distributed).

The `LimitRequestFields` directive allows the server administrator to modify the limit on the number of request header fields allowed in an HTTP request. A server needs this value to be larger than the number of fields that a normal client request might include. The number of request header fields used by a client rarely exceeds 20, but this may vary among different client implementations, often depending upon the extent to which a user has configured their browser to support detailed content negotiation. Optional HTTP extensions are often expressed using request header fields.

This directive gives the server administrator greater control over abnormal client request behavior, which may be useful for avoiding some forms of denial-of-service attacks. The value should be increased if normal clients see an error response from the server that indicates too many fields were sent in the request.

For example:

```
LimitRequestFields 50
```

**Warning**

When name-based virtual hosting is used, the value for this directive is taken from the default (first-listed) virtual host for the local IP and port combination.



<b>Description:</b>	Limits the size of the HTTP request header allowed from the client
<b>Syntax:</b>	<code>LimitRequestFieldSize</code> <i>bytes</i>
<b>Default:</b>	<code>LimitRequestFieldSize</code> 8190
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

This directive specifies the number of *bytes* that will be allowed in an HTTP request header.

The `LimitRequestFieldSize` directive allows the server administrator to set the limit on the allowed size of an HTTP request header field. A server needs this value to be large enough to hold any one header field from a normal client request. The size of a normal request header field will vary greatly among different client implementations, often depending upon the extent to which a user has configured their browser to support detailed content negotiation. SPNEGO authentication headers can be up to 12392 bytes.

This directive gives the server administrator greater control over abnormal client request behavior, which may be useful for avoiding some forms of denial-of-service attacks.

For example:

```
LimitRequestFieldSize 4094
```

Under normal conditions, the value should not be changed from the default.

### **Warning**

When name-based virtual hosting is used, the value for this directive is taken from the default (first-listed) virtual host best matching the current IP address and port combination.



<b>Description:</b>	Limit the size of the HTTP request line that will be accepted from the client
<b>Syntax:</b>	<code>LimitRequestLine bytes</code>
<b>Default:</b>	<code>LimitRequestLine 8190</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

This directive sets the number of *bytes* that will be allowed on the HTTP request-line.

The `LimitRequestLine` directive allows the server administrator to set the limit on the allowed size of a client's HTTP request-line. Since the request-line consists of the HTTP method, URI, and protocol version, the `LimitRequestLine` directive places a restriction on the length of a request-URI allowed for a request on the server. A server needs this value to be large enough to hold any of its resource names, including any information that might be passed in the query part of a GET request.

This directive gives the server administrator greater control over abnormal client request behavior, which may be useful for avoiding some forms of denial-of-service attacks.

For example:

```
LimitRequestLine 4094
```

Under normal conditions, the value should not be changed from the default.

**Warning**

When name-based virtual hosting is used, the value for this directive is taken from the default (first-listed) virtual host best matching the current IP address and port combination.



## LimitXMLRequestBody Directive

<b>Description:</b>	Limits the size of an XML-based request body
<b>Syntax:</b>	<code>LimitXMLRequestBody bytes</code>
<b>Default:</b>	<code>LimitXMLRequestBody 1000000</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

Limit (in bytes) on maximum size of an XML-based request body. A value of 0 will disable any checking.

Example:

```
LimitXMLRequestBody 0
```



**Description:** Applies the enclosed directives only to matching URLs

**Syntax:** `<Location URL-path|URL> ...  
</Location>`

**Context:** server config, virtual host

**Status:** Core

**Module:** core

The `<Location>` directive limits the scope of the enclosed directives by URL. It is similar to the `<Directory>` directive, and starts a subsection which is terminated with a `</Location>` directive. `<Location>` sections are processed in the order they appear in the configuration file, after the `<Directory>` sections and `.htaccess` files are read, and after the `<Files>` sections.

`<Location>` sections operate completely outside the filesystem. This has several consequences. Most importantly, `<Location>` directives should not be used to control access to filesystem locations. Since several different URLs may map to the same filesystem location, such access controls may be circumvented.

The enclosed directives will be applied to the request if the path component of the URL meets *any* of the following criteria:

- The specified location matches exactly the path component of the URL.
- The specified location, which ends in a forward slash, is a prefix of the path component of the URL (treated as a context root).
- The specified location, with the addition of a trailing slash, is a prefix of the path component of the URL (also treated as a context root).

In the example below, where no trailing slash is used, requests to /private1, /private1/ and /private1/file.txt will have the enclosed directives applied, but /private1other would not.

```
<Location "/private1">  
    # ...  
</Location>
```

In the example below, where a trailing slash is used, requests to /private2/ and /private2/file.txt will have the enclosed directives applied, but /private2 and /private2other would not.

```
<Location "/private2/">  
    # ...  
</Location>
```

### When to use `<Location>`

Use `<Location>` to apply directives to content that lives outside the filesystem. For content that lives in the filesystem, use `<Directory>` and `<Files>`. An exception is `<Location "/">`, which is an easy way to apply a configuration to the entire server.

For all origin (non-proxy) requests, the URL to be matched is a URL-path of the form /path/. *No scheme, hostname, port, or query string may be included.* For proxy requests, the URL to be matched is of the form scheme://servername/path, and you must include the prefix.

The URL may use wildcards. In a wild-card string, ? matches any single character, and \* matches any sequences of characters. Neither wildcard character matches a / in the URL-path.

[Regular expressions](#) can also be used, with the addition of the ~ character. For example:

```
<Location ~ "/(extra|special)/data">
    #...
</Location>
```

would match URLs that contained the substring /extra/data or /special/data. The directive [<LocationMatch>](#) behaves identical to the regex version of [<Location>](#), and is preferred, for the simple reason that ~ is hard to distinguish from - in many fonts.

The [<Location>](#) functionality is especially useful when combined with the [SetHandler](#) directive. For example, to enable status requests but allow them only from browsers at example.com, you might use:

```
<Location "/status">
    SetHandler server-status
    Require host example.com
</Location>
```

### Note about / (slash)

The slash character has special meaning depending on where in a URL it appears. People may be used to its behavior in the filesystem where multiple adjacent slashes are frequently collapsed to a single slash (*i.e.*, /home///foo is the same as /home/foo). In URL-space this is not necessarily true. The [<LocationMatch>](#) directive and the regex version of [<Location>](#) require you to explicitly specify multiple slashes if that is your intention.

For example, `<LocationMatch "/abc">` would match the request URL `/abc` but not the request URL `//abc`. The (non-regex) `<Location>` directive behaves similarly when used for proxy requests. But when (non-regex) `<Location>` is used for non-proxy requests it will implicitly match multiple slashes with a single slash. For example, if you specify `<Location "/abc/def">` and the request is to `/abc//def` then it will match.

## See also

- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received.
- [LocationMatch](#)



**Description:** Applies the enclosed directives only to regular-expression matching URLs

**Syntax:** <LocationMatch *regex*> ...  
</LocationMatch>

**Context:** server config, virtual host

**Status:** Core

**Module:** core

The <LocationMatch> directive limits the scope of the enclosed directives by URL, in an identical manner to <Location>.

However, it takes a [regular expression](#) as an argument instead of a simple string. For example:

```
<LocationMatch "/(extra|special)/data">  
    # ...  
</LocationMatch>
```

would match URLs that contained the substring /extra/data or /special/data.

If the intent is that a URL **starts with** /extra/data, rather than merely **contains** /extra/data, prefix the regular expression with a ^ to require this.

```
<LocationMatch "^/(extra|special)/data">
```

From 2.4.8 onwards, named groups and backreferences are captured and written to the environment with the corresponding name prefixed with "MATCH\_" and in upper case. This allows elements of URLs to be referenced from within [expressions](#) and modules like [mod\\_rewrite](#). In order to prevent confusion,

numbered (unnamed) backreferences are ignored. Use named groups instead.

```
<LocationMatch "^(/combined/(?<sitename>[^/]-  
    require ldap-group cn=%{env:MATCH_SITENAME}  
</LocationMatch>
```

## See also

- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received



## LogLevel

<b>Description:</b>	Controls the verbosity of the ErrorLog
<b>Syntax:</b>	LogLevel [ <i>module:</i> ]level [ <i>module:level</i> ] ...
<b>Default:</b>	LogLevel warn
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Per-module and per-directory configuration is available in Apache HTTP Server 2.3.6 and later

**LogLevel** adjusts the verbosity of the messages recorded in the error logs (see **ErrorLog** directive). The following *levels* are available, in order of decreasing significance:

Level	Description	Example
emerg	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
alert	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."

debug	Debug-level messages	"Opening config file ..."
trace1	Trace messages	"proxy: FTP: control connection complete"
trace2	Trace messages	"proxy: CONNECT: sending the CONNECT request to the remote proxy"
trace3	Trace messages	"openssl: Handshake: start"
trace4	Trace messages	"read from buffered SSL brigade, mode 0, 17 bytes"
trace5	Trace messages	"map lookup FAILED: map=rewritemap key=keyname"
trace6	Trace messages	"cache lookup FAILED, forcing new map lookup"
trace7	Trace messages, dumping large amounts of data	"  0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15  "
trace8	Trace messages, dumping large amounts of data	"  0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15  "

When a particular level is specified, messages from all other levels of higher significance will be reported as well. *E.g.*, when `LogLevel info` is specified, then messages with log levels of `notice` and `warn` will also be posted.

Using a level of at least `crit` is recommended.

For example:

```
LogLevel notice
```

## Note

When logging to a regular file, messages of the level `notice` cannot be suppressed and thus are always logged. However, this doesn't apply when logging is done using `syslog`.

Specifying a level without a module name will reset the level for all modules to that level. Specifying a level with a module name will set the level for that module only. It is possible to use the module source file name, the module identifier, or the module identifier with the trailing `_module` omitted as module specification. This means the following three specifications are equivalent:

```
LogLevel info ssl:warn
LogLevel info mod_ssl.c:warn
LogLevel info ssl_module:warn
```

It is also possible to change the level per directory:

```
LogLevel info
<Directory "/usr/local/apache/htdocs/app">
  LogLevel debug
</Directory>
```

Per directory loglevel configuration only affects messages that are logged after the request has been parsed and that are associated with the request. Log messages which are associated with the connection or the server are not affected.

## See also

- [ErrorLog](#)
- [ErrorLogFormat](#)
- [Apache HTTP Server Log Files](#)



## MaxKeepAliveRequests Directive

<b>Description:</b>	Number of requests allowed on a persistent connection
<b>Syntax:</b>	MaxKeepAliveRequests <i>number</i>
<b>Default:</b>	MaxKeepAliveRequests 100
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The `MaxKeepAliveRequests` directive limits the number of requests allowed per connection when `KeepAlive` is on. If it is set to 0, unlimited requests will be allowed. We recommend that this setting be kept to a high value for maximum server performance.

For example:

```
MaxKeepAliveRequests 500
```



<b>Description:</b>	Number of overlapping ranges (eg: 100-200, 150-300) allowed before returning the complete resource
<b>Syntax:</b>	MaxRangeOverlaps default   unlimited   none   <i>number-of-ranges</i>
<b>Default:</b>	MaxRangeOverlaps 20
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.15 and later

The `MaxRangeOverlaps` directive limits the number of overlapping HTTP ranges the server is willing to return to the client. If more overlapping ranges than permitted are requested, the complete resource is returned instead.

### default

Limits the number of overlapping ranges to a compile-time default of 20.

### none

No overlapping Range headers are allowed.

### unlimited

The server does not limit the number of overlapping ranges it is willing to satisfy.

### *number-of-ranges*

A positive number representing the maximum number of overlapping ranges the server is willing to satisfy.



## MaxRangeReversals Directive

<b>Description:</b>	Number of range reversals (eg: 100-200, 50-70) allowed before returning the complete resource
<b>Syntax:</b>	MaxRangeReversals default   unlimited   none   <i>number-of-ranges</i>
<b>Default:</b>	MaxRangeReversals 20
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.15 and later

The `MaxRangeReversals` directive limits the number of HTTP Range reversals the server is willing to return to the client. If more ranges reversals than permitted are requested, the complete resource is returned instead.

### default

Limits the number of range reversals to a compile-time default of 20.

### none

No Range reversals headers are allowed.

### unlimited

The server does not limit the number of range reversals it is willing to satisfy.

### *number-of-ranges*

A positive number representing the maximum number of range reversals the server is willing to satisfy.



## MaxRanges Directive

<b>Description:</b>	Number of ranges allowed before returning the complete resource
<b>Syntax:</b>	MaxRanges default   unlimited   none   <i>number-of-ranges</i>
<b>Default:</b>	MaxRanges 200
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.15 and later

The **MaxRanges** directive limits the number of HTTP ranges the server is willing to return to the client. If more ranges than permitted are requested, the complete resource is returned instead.

### **default**

Limits the number of ranges to a compile-time default of 200.

### **none**

Range headers are ignored.

### **unlimited**

The server does not limit the number of ranges it is willing to satisfy.

### ***number-of-ranges***

A positive number representing the maximum number of ranges the server is willing to satisfy.



## MergeTrailers Directive

<b>Description:</b>	Determines whether trailers are merged into headers
<b>Syntax:</b>	MergeTrailers [on off]
<b>Default:</b>	MergeTrailers off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	2.4.11 and later

This directive controls whether HTTP trailers are copied into the internal representation of HTTP headers. This merging occurs when the request body has been completely consumed, long after most header processing would have a chance to examine or modify request headers.

This option is provided for compatibility with releases prior to 2.4.11, where trailers were always merged.



<b>Description:</b>	Configures mutex mechanism and lock file directory for all or specified mutexes
<b>Syntax:</b>	Mutex <i>mechanism</i> [default  <i>mutex-name</i> ] ... [OmitPID]
<b>Default:</b>	Mutex default
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.4 and later

The **Mutex** directive sets the mechanism, and optionally the lock file location, that httpd and modules use to serialize access to resources. Specify default as the second argument to change the settings for all mutexes; specify a mutex name (see table below) as the second argument to override defaults only for that mutex.

The **Mutex** directive is typically used in the following exceptional situations:

- change the mutex mechanism when the default mechanism selected by **APR** has a functional or performance problem
- change the directory used by file-based mutexes when the default directory does not support locking

### Supported modules

This directive only configures mutexes which have been registered with the core server using the `ap_mutex_register()` API. All modules bundled with httpd support the **Mutex** directive, but third-party modules may not. Consult the documentation of the third-party module, which must indicate the mutex name(s) which can be configured if this

directive is supported.

The following mutex *mechanisms* are available:

- `default` | `yes`  
This selects the default locking implementation, as determined by [APR](#). The default locking implementation can be displayed by running [httpd](#) with the `-V` option.
- `none` | `no`  
This effectively disables the mutex, and is only allowed for a mutex if the module indicates that it is a valid choice. Consult the module documentation for more information.
- `posixsem`  
This is a mutex variant based on a Posix semaphore.

#### **Warning**

The semaphore ownership is not recovered if a thread in the process holding the mutex segfaults, resulting in a hang of the web server.

- `sysvsem`  
This is a mutex variant based on a SystemV IPC semaphore.

#### **Warning**

It is possible to "leak" SysV semaphores if processes crash before the semaphore is removed.

#### **Security**

The semaphore API allows for a denial of service attack by any CGIs running under the same uid as the webserver

(i.e., all CGIs, unless you use something like [suexec](#) or [cgiwrapper](#)).

- **sem**  
This selects the "best" available semaphore implementation, choosing between Posix and SystemV IPC semaphores, in that order.
- **pthread**  
This is a mutex variant based on cross-process Posix thread mutexes.

### **Warning**

On most systems, if a child process terminates abnormally while holding a mutex that uses this implementation, the server will deadlock and stop responding to requests. When this occurs, the server will require a manual restart to recover.

Solaris and Linux are notable exceptions as they provide a mechanism which usually allows the mutex to be recovered after a child process terminates abnormally while holding a mutex.

If your system is POSIX compliant or if it implements the `pthread_mutexattr_setrobust_np()` function, you may be able to use the `pthread` option safely.

- **fcntl:/path/to/mutex**  
This is a mutex variant where a physical (lock-)file and the `fcntl()` function are used as the mutex.

### **Warning**

When multiple mutexes based on this mechanism are used within multi-threaded, multi-process environments, deadlock errors (EDEADLK) can be reported for valid mutex operations if `fcntl()` is not thread-aware, such as on Solaris.

- `flock:/path/to/mutex`  
This is similar to the `fcntl:/path/to/mutex` method with the exception that the `flock()` function is used to provide file locking.
- `file:/path/to/mutex`  
This selects the "best" available file locking implementation, choosing between `fcntl` and `flock`, in that order.

Most mechanisms are only available on selected platforms, where the underlying platform and [APR](#) support it. Mechanisms which aren't available on all platforms are *posixsem*, *sysvsem*, *sem*, *pthread*, *fcntl*, *flock*, and *file*.

With the file-based mechanisms *fcntl* and *flock*, the path, if provided, is a directory where the lock file will be created. The default directory is httpd's run-time file directory relative to [ServerRoot](#). Always use a local disk filesystem for `/path/to/mutex` and never a directory residing on a NFS- or AFS-filesystem. The basename of the file will be the mutex type, an optional instance string provided by the module, and unless the `OmitPID` keyword is specified, the process id of the httpd parent process will be appended to make the file name unique, avoiding conflicts when multiple httpd instances share a lock file directory. For example, if the mutex name is `mpm-accept` and the lock file directory is `/var/httpd/locks`, the lock file name for the httpd instance with parent process id 12345 would be `/var/httpd/locks/mpm-accept.12345`.

## Security

It is best to *avoid* putting mutex files in a world-writable directory such as `/var/tmp` because someone could create a denial of service attack and prevent the server from starting by creating a lockfile with the same name as the one the server will try to create.

The following table documents the names of mutexes used by `httpd` and bundled modules.

Mutex name	Module(s)	Protected resource
<code>mpm-accept</code>	<a href="#"><code>prefork</code></a> and <a href="#"><code>worker</code></a> MPMs	incoming connections, to avoid the thundering herd problem; for more information, refer to the <a href="#">performance tuning</a> documentation
<code>authdigest-client</code>	<a href="#"><code>mod_auth_digest</code></a>	client list in shared memory
<code>authdigest-opaque</code>	<a href="#"><code>mod_auth_digest</code></a>	counter in shared memory
<code>ldap-cache</code>	<a href="#"><code>mod_ldap</code></a>	LDAP result cache
<code>rewrite-map</code>	<a href="#"><code>mod_rewrite</code></a>	communication with external mapping programs, to avoid intermixed I/O from multiple requests
<code>ssl-cache</code>	<a href="#"><code>mod_ssl</code></a>	SSL session cache
<code>ssl-stapling</code>	<a href="#"><code>mod_ssl</code></a>	OCSP stapling response cache
<code>watchdog-callback</code>	<a href="#"><code>mod_watchdog</code></a>	callback function of a particular client module

The `OmitPID` keyword suppresses the addition of the `httpd` parent process id from the lock file name.

In the following example, the mutex mechanism for the MPM `accept` mutex will be changed from the compiled-in default to `fcntl`, with the associated lock file created in directory `/var/httpd/locks`. The mutex mechanism for all other mutexes will be changed from the compiled-in default to `sysvsem`.

```
Mutex sysvsem default
Mutex fcntl:/var/httpd/locks mpm-accept
```



**Description:** DEPRECATED: Designates an IP address for name-virtual hosting

**Syntax:** `NameVirtualHost addr[:port]`

**Context:** server config

**Status:** Core

**Module:** core

Prior to 2.3.11, `NameVirtualHost` was required to instruct the server that a particular IP address and port combination was usable as a name-based virtual host. In 2.3.11 and later, any time an IP address and port combination is used in multiple virtual hosts, name-based virtual hosting is automatically enabled for that address.

This directive currently has no effect.

## See also

- [Virtual Hosts documentation](#)



<b>Description:</b>	Configures what features are available in a particular directory
<b>Syntax:</b>	Options [+ -] <i>option</i> [[+ -] <i>option</i> ] ...
<b>Default:</b>	Options FollowSymLinks
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	The default was changed from All to FollowSymLinks in 2.3.11

The **Options** directive controls which server features are available in a particular directory.

*option* can be set to None, in which case none of the extra features are enabled, or one or more of the following:

### All

All options except for MultiViews.

### ExecCGI

Execution of CGI scripts using [mod\\_cgi](#) is permitted.

### FollowSymLinks

The server will follow symbolic links in this directory. This is the default setting.

Even though the server follows the symlink it does *not* change the pathname used to match against [<Directory>](#) sections.

The FollowSymLinks and SymLinksIfOwnerMatch

[Options](#) work only in [<Directory>](#) sections or .htaccess files.

Omitting this option should not be considered a security restriction, since symlink testing is subject to race conditions that make it circumventable.

## **Includes**

Server-side includes provided by [mod\\_include](#) are permitted.

## **IncludesNOEXEC**

Server-side includes are permitted, but the `#exec cmd` and `#exec cgi` are disabled. It is still possible to `#include` virtual CGI scripts from [ScriptAliased](#) directories.

## **Indexes**

If a URL which maps to a directory is requested and there is no [DirectoryIndex](#) (e.g., `index.html`) in that directory, then [mod\\_autoindex](#) will return a formatted listing of the directory.

## **MultiViews**

[Content negotiated](#) "MultiViews" are allowed using [mod\\_negotiation](#).

### **Note**

This option gets ignored if set anywhere other than [<Directory>](#), as [mod\\_negotiation](#) needs real resources to compare against and evaluate from.

## **SymLinksIfOwnerMatch**

The server will only follow symbolic links for which the target file or directory is owned by the same user id as the link.

## Note

The `FollowSymLinks` and `SymLinksIfOwnerMatch` [Options](#) work only in `<Directory>` sections or `.htaccess` files.

This option should not be considered a security restriction, since symlink testing is subject to race conditions that make it circumventable.

Normally, if multiple [Options](#) could apply to a directory, then the most specific one is used and others are ignored; the options are not merged. (See [how sections are merged](#).) However if *all* the options on the [Options](#) directive are preceded by a `+` or `-` symbol, the options are merged. Any options preceded by a `+` are added to the options currently in force, and any options preceded by a `-` are removed from the options currently in force.

## Note

Mixing [Options](#) with a `+` or `-` with those without is not valid syntax and will be rejected during server startup by the syntax check with an abort.

For example, without any `+` and `-` symbols:

```
<Directory "/web/docs">
  Options Indexes FollowSymLinks
</Directory>

<Directory "/web/docs/spec">
  Options Includes
</Directory>
```

then only `Includes` will be set for the `/web/docs/spec`

directory. However if the second `Options` directive uses the + and - symbols:

```
<Directory "/web/docs">
  Options Indexes FollowSymLinks
</Directory>

<Directory "/web/docs/spec">
  Options +Includes -Indexes
</Directory>
```

then the options `FollowSymLinks` and `Includes` are set for the `/web/docs/spec` directory.

#### **Note**

Using `-IncludesNOEXEC` or `-Includes` disables server-side includes completely regardless of the previous setting.

The default in the absence of any other settings is `FollowSymLinks`.



<b>Description:</b>	Protocol for a listening socket
<b>Syntax:</b>	Protocol <i>protocol</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache 2.1.5 and later. On Windows, from Apache 2.3.3 and later.

This directive specifies the protocol used for a specific listening socket. The protocol is used to determine which module should handle a request and to apply protocol specific optimizations with the [AcceptFilter](#) directive.

You only need to set the protocol if you are running on non-standard ports; otherwise, `http` is assumed for port 80 and `https` for port 443.

For example, if you are running `https` on a non-standard port, specify the protocol explicitly:

```
Protocol https
```

You can also specify the protocol using the [Listen](#) directive.

## See also

- [AcceptFilter](#)
- [Listen](#)



<b>Description:</b>	Protocols available for a server/virtual host
<b>Syntax:</b>	Protocols <i>protocol</i> ...
<b>Default:</b>	Protocols http/1.1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Only available from Apache 2.4.17 and later.

This directive specifies the list of protocols supported for a server/virtual host. The list determines the allowed protocols a client may negotiate for this server/host.

You need to set protocols if you want to extend the available protocols for a server/host. By default, only the http/1.1 protocol (which includes the compatibility with 1.0 and 0.9 clients) is allowed.

For example, if you want to support HTTP/2 for a server with TLS, specify:

```
Protocols h2 http/1.1
```

Valid protocols are http/1.1 for http and https connections, h2 on https connections and h2c for http connections. Modules may enable more protocols.

It is safe to specify protocols that are unavailable/disabled. Such protocol names will simply be ignored.

Protocols specified in base servers are inherited for virtual hosts only if the virtual host has no own Protocols directive. Or, the other way around, Protocols directives in virtual hosts replace any such directive in the base server.

## See also

- [ProtocolsHonorOrder](#)



<b>Description:</b>	Determines if order of Protocols determines precedence during negotiation
<b>Syntax:</b>	ProtocolsHonorOrder On Off
<b>Default:</b>	ProtocolsHonorOrder On
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Only available from Apache 2.4.17 and later.

This directive specifies if the server should honor the order in which the `Protocols` directive lists protocols.

If configured Off, the client supplied list order of protocols has precedence over the order in the server configuration.

With `ProtocolsHonorOrder` set to on (default), the client ordering does not matter and only the ordering in the server settings influences the outcome of the protocol negotiation.

## See also

- [Protocols](#)



## QualifyRedirectURL Directive

<b>Description:</b>	Controls whether the REDIRECT_URL environment variable is fully qualified
<b>Syntax:</b>	QualifyRedirectURL ON OFF
<b>Default:</b>	QualifyRedirectURL OFF
<b>Context:</b>	server config, virtual host, directory
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Directive supported in 2.4.18 and later. 2.4.17 acted as if 'QualifyRedirectURL ON' was configured.

This directive controls whether the server will ensure that the REDIRECT\_URL environment variable is fully qualified. By default, the variable contains the verbatim URL requested by the client, such as "/index.html". With [QualifyRedirectURL ON](#), the same request would result in a value such as "http://www.example.com/index.html".

Even without this directive set, when a request is issued against a fully qualified URL, REDIRECT\_URL will remain fully qualified.



## RegisterHttpMethod Directive

<b>Description:</b>	Register non-standard HTTP methods
<b>Syntax:</b>	<code>RegisterHttpMethod <i>method</i> [<i>method</i> [...]]</code>
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

HTTP Methods that are not conforming to the relevant RFCs are normally rejected by request processing in Apache HTTPD. To avoid this, modules can register non-standard HTTP methods they support. The `RegisterHttpMethod` allows to register such methods manually. This can be useful for if such methods are forwarded for external processing, e.g. to a CGI script.



<b>Description:</b>	Limits the CPU consumption of processes launched by Apache httpd children
<b>Syntax:</b>	<code>RLimitCPU seconds max [seconds max]</code>
<b>Default:</b>	Unset; uses operating system defaults
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

Takes 1 or 2 parameters. The first parameter sets the soft resource limit for all processes and the second parameter sets the maximum resource limit. Either parameter can be a number, or max to indicate to the server that the limit should be set to the maximum allowed by the operating system configuration. Raising the maximum resource limit requires that the server is running as root or in the initial startup phase.

This applies to processes forked from Apache httpd children servicing requests, not the Apache httpd children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked from the Apache httpd parent, such as piped logs.

CPU resource limits are expressed in seconds per process.

## See also

- [RLimitMEM](#)
- [RLimitNPROC](#)



<b>Description:</b>	Limits the memory consumption of processes launched by Apache httpd children
<b>Syntax:</b>	RLimitMEM <i>bytes max</i> [ <i>bytes max</i> ]
<b>Default:</b>	Unset; uses operating system defaults
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

Takes 1 or 2 parameters. The first parameter sets the soft resource limit for all processes and the second parameter sets the maximum resource limit. Either parameter can be a number, or max to indicate to the server that the limit should be set to the maximum allowed by the operating system configuration. Raising the maximum resource limit requires that the server is running as root or in the initial startup phase.

This applies to processes forked from Apache httpd children servicing requests, not the Apache httpd children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked from the Apache httpd parent, such as piped logs.

Memory resource limits are expressed in bytes per process.

## See also

- [RLimitCPU](#)
- [RLimitNPROC](#)



<b>Description:</b>	Limits the number of processes that can be launched by processes launched by Apache httpd children
<b>Syntax:</b>	<code>RLimitNPROC <i>number</i>   max [<i>number</i>   max]</code>
<b>Default:</b>	Unset; uses operating system defaults
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

Takes 1 or 2 parameters. The first parameter sets the soft resource limit for all processes, and the second parameter sets the maximum resource limit. Either parameter can be a number, or max to indicate to the server that the limit should be set to the maximum allowed by the operating system configuration. Raising the maximum resource limit requires that the server is running as root or in the initial startup phase.

This applies to processes forked from Apache httpd children servicing requests, not the Apache httpd children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked from the Apache httpd parent, such as piped logs.

Process limits control the number of processes per user.

### Note

If CGI processes are **not** running under user ids other than the web server user id, this directive will limit the number of processes that the server itself can create. Evidence of this situation will be indicated by **cannot fork** messages in the `error_log`.

## See also

- [RLimitMEM](#)
- [RLimitCPU](#)



<b>Description:</b>	Technique for locating the interpreter for CGI scripts
<b>Syntax:</b>	ScriptInterpreterSource Registry Registry-Strict Script
<b>Default:</b>	ScriptInterpreterSource Script
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Win32 only.

This directive is used to control how Apache httpd finds the interpreter used to run CGI scripts. The default setting is `Script`. This causes Apache httpd to use the interpreter pointed to by the shebang line (first line, starting with `#!`) in the script. On Win32 systems this line usually looks like:

```
#!C:/Perl/bin/perl.exe
```

or, if `perl` is in the PATH, simply:

```
#!perl
```

Setting `ScriptInterpreterSource Registry` will cause the Windows Registry tree `HKEY_CLASSES_ROOT` to be searched using the script file extension (e.g., `.pl`) as a search key. The command defined by the registry subkey `Shell\ExecCGI\Command` or, if it does not exist, by the subkey `Shell\Open\Command` is used to open the script file. If the registry keys cannot be found, Apache httpd falls back to the behavior of the `Script` option.

## Security

Be careful when using `ScriptInterpreterSource` Registry with `ScriptAlias`'ed directories, because Apache httpd will try to execute **every** file within this directory. The Registry setting may cause undesired program calls on files which are typically not executed. For example, the default open command on `.htm` files on most Windows systems will execute Microsoft Internet Explorer, so any HTTP request for an `.htm` file existing within the script directory would start the browser in the background on the server. This is a good way to crash your system within a minute or so.

The option `Registry-Strict` which is new in Apache HTTP Server 2.0 does the same thing as `Registry` but uses only the subkey `Shell\ExecCGI\Command`. The `ExecCGI` key is not a common one. It must be configured manually in the windows registry and hence prevents accidental program calls on your system.



<b>Description:</b>	Determine if mod_status displays the first 63 characters of a request or the last 63, assuming the request itself is greater than 63 chars.
<b>Syntax:</b>	SeeRequestTail On Off
<b>Default:</b>	SeeRequestTail Off
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache httpd 2.2.7 and later.

mod\_status with ExtendedStatus On displays the actual request being handled. For historical purposes, only 63 characters of the request are actually stored for display purposes. This directive controls whether the 1st 63 characters are stored (the previous behavior and the default) or if the last 63 characters are. This is only applicable, of course, if the length of the request is 64 characters or greater.

If Apache httpd is handling  
GET /disk1/storage/apache/htdocs/images/imagestore1/
mod\_status displays as follows:

<b>Off (default)</b>	GET /disk1/storage/apache/htdocs/images/imagestore1/
<b>On</b>	orage/apache/htdocs/images/imagestore1/food/apples.jp



<b>Description:</b>	Email address that the server includes in error messages sent to the client
<b>Syntax:</b>	ServerAdmin <i>email-address</i>  URL
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The `ServerAdmin` sets the contact address that the server includes in any error messages it returns to the client. If the `httpd` doesn't recognize the supplied argument as an URL, it assumes, that it's an *email-address* and prepends it with `mailto:` in hyperlink targets. However, it's recommended to actually use an email address, since there are a lot of CGI scripts that make that assumption. If you want to use an URL, it should point to another server under your control. Otherwise users may not be able to contact you in case of errors.

It may be worth setting up a dedicated address for this, e.g.

```
ServerAdmin www-admin@foo.example.com
```

as users do not always mention that they are talking about the server!



<b>Description:</b>	Alternate names for a host used when matching requests to name-virtual hosts
<b>Syntax:</b>	<code>ServerAlias <i>hostname</i> [<i>hostname</i>] ...</code>
<b>Context:</b>	virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The `ServerAlias` directive sets the alternate names for a host, for use with [name-based virtual hosts](#). The `ServerAlias` may include wildcards, if appropriate.

```
<VirtualHost *:80>
  ServerName server.example.com
  ServerAlias server server2.example.com se
  ServerAlias *.example.com
  UseCanonicalName Off
  # ...
</VirtualHost>
```

Name-based virtual hosts for the best-matching set of `<virtualhost>`s are processed in the order they appear in the configuration. The first matching `ServerName` or `ServerAlias` is used, with no different precedence for wildcards (nor for `ServerName` vs. `ServerAlias`).

The complete list of names in the `<VirtualHost>` directive are treated just like a (non wildcard) `ServerAlias`.

## See also

- [UseCanonicalName](#)
- [Apache HTTP Server Virtual Host documentation](#)



<b>Description:</b>	Hostname and port that the server uses to identify itself
<b>Syntax:</b>	<code>ServerName [scheme://]domain-name ip-address[:port]</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The `ServerName` directive sets the request scheme, hostname and port that the server uses to identify itself.

`ServerName` is used (possibly in conjunction with `ServerAlias`) to uniquely identify a virtual host, when using [name-based virtual hosts](#).

Additionally, this is used when creating self-referential redirection URLs when `UseCanonicalName` is set to a non-default value.

For example, if the name of the machine hosting the web server is `simple.example.com`, but the machine also has the DNS alias `www.example.com` and you wish the web server to be so identified, the following directive should be used:

```
ServerName www.example.com
```

The `ServerName` directive may appear anywhere within the definition of a server. However, each appearance overrides the previous appearance (within that server).

If no `ServerName` is specified, the server attempts to deduce the client visible hostname by first asking the operating system for the system hostname, and if that fails, performing a reverse lookup on an IP address present on the system.

If no port is specified in the `ServerName`, then the server will use the port from the incoming request. For optimal reliability and predictability, you should specify an explicit hostname and port using the `ServerName` directive.

If you are using [name-based virtual hosts](#), the `ServerName` inside a `<VirtualHost>` section specifies what hostname must appear in the request's `Host` : header to match this virtual host.

Sometimes, the server runs behind a device that processes SSL, such as a reverse proxy, load balancer or SSL offload appliance. When this is the case, specify the `https://` scheme and the port number to which the clients connect in the `ServerName` directive to make sure that the server generates the correct self-referential URLs.

See the description of the `UseCanonicalName` and `UseCanonicalPhysicalPort` directives for settings which determine whether self-referential URLs (e.g., by the `mod_dir` module) will refer to the specified port, or to the port number given in the client's request.

Failure to set `ServerName` to a name that your server can resolve to an IP address will result in a startup warning. `httpd` will then use whatever hostname it can determine, using the system's `hostname` command. This will almost never be the hostname you actually want.

```
httpd: Could not reliably determine the server's fully
qualified domain name, using rocinante.local for
ServerName
```

## See also

- [Issues Regarding DNS and Apache HTTP Server](#)
- [Apache HTTP Server virtual host documentation](#)
- [UseCanonicalName](#)
- [UseCanonicalPhysicalPort](#)
- [ServerAlias](#)



<b>Description:</b>	Legacy URL pathname for a name-based virtual host that is accessed by an incompatible browser
<b>Syntax:</b>	<code>ServerPath</code> <i>URL-path</i>
<b>Context:</b>	virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The `ServerPath` directive sets the legacy URL pathname for a host, for use with [name-based virtual hosts](#).

## See also

- [Apache HTTP Server Virtual Host documentation](#)



<b>Description:</b>	Base directory for the server installation
<b>Syntax:</b>	ServerRoot <i>directory-path</i>
<b>Default:</b>	ServerRoot /usr/local/apache
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

The **ServerRoot** directive sets the directory in which the server lives. Typically it will contain the subdirectories `conf/` and `logs/`. Relative paths in other configuration directives (such as **Include** or **LoadModule**, for example) are taken as relative to this directory.

```
ServerRoot "/home/httpd"
```

The default location of **ServerRoot** may be modified by using the `--prefix` argument to **configure**, and most third-party distributions of the server have a different default location from the one listed above.

## See also

- [the `-d` option to `httpd`](#)
- [the security tips](#) for information on how to properly set permissions on the **ServerRoot**



<b>Description:</b>	Configures the footer on server-generated documents
<b>Syntax:</b>	ServerSignature On Off EMail
<b>Default:</b>	ServerSignature Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	core

The [ServerSignature](#) directive allows the configuration of a trailing footer line under server-generated documents (error messages, [mod\\_proxy](#) ftp directory listings, [mod\\_info](#) output, ...). The reason why you would want to enable such a footer line is that in a chain of proxies, the user often has no possibility to tell which of the chained servers actually produced a returned error message.

The Off setting, which is the default, suppresses the footer line (and is therefore compatible with the behavior of Apache-1.2 and below). The On setting simply adds a line with the server version number and [ServerName](#) of the serving virtual host, and the EMail setting additionally creates a "mailto:" reference to the [ServerAdmin](#) of the referenced document.

After version 2.0.44, the details of the server version number presented are controlled by the [ServerTokens](#) directive.

## See also

- [ServerTokens](#)



<b>Description:</b>	Configures the Server HTTP response header
<b>Syntax:</b>	ServerTokens Major Minor Min[imal] Prod[uctOnly] OS
<b>Default:</b>	ServerTokens Full
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

This directive controls whether Server response header field which is sent back to clients includes a description of the generic OS-type of the server as well as information about compiled-in modules.

#### **ServerTokens Full (or not specified)**

Server sends (e.g.): Server: Apache/2.4.2 (Unix)  
PHP/4.2.2 MyMod/1.2

#### **ServerTokens Prod[uctOnly]**

Server sends (e.g.): Server: Apache

#### **ServerTokens Major**

Server sends (e.g.): Server: Apache/2

#### **ServerTokens Minor**

Server sends (e.g.): Server: Apache/2.4

#### **ServerTokens Min[imal]**

Server sends (e.g.): Server: Apache/2.4.2

#### **ServerTokens OS**

Server sends (e.g.): Server: Apache/2.4.2 (Unix)

This setting applies to the entire server, and cannot be enabled or disabled on a virtualhost-by-virtualhost basis.

After version 2.0.44, this directive also controls the information

presented by the [ServerSignature](#) directive.

Setting `ServerTokens` to less than `minimal` is not recommended because it makes it more difficult to debug interoperational problems. Also note that disabling the `Server:` header does nothing at all to make your server more secure. The idea of "security through obscurity" is a myth and leads to a false sense of safety.

## See also

- [ServerSignature](#)



<b>Description:</b>	Forces all matching files to be processed by a handler
<b>Syntax:</b>	SetHandler <i>handler-name</i>   none   <i>expression</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	expression argument 2.4.19 and later

When placed into an .htaccess file or a [<Directory>](#) or [<Location>](#) section, this directive forces all matching files to be parsed through the [handler](#) given by *handler-name*. For example, if you had a directory you wanted to be parsed entirely as imagemap rule files, regardless of extension, you might put the following into an .htaccess file in that directory:

```
SetHandler imap-file
```

Another example: if you wanted to have the server display a status report whenever a URL of `http://servername/status` was called, you might put the following into `httpd.conf`:

```
<Location "/status">  
    SetHandler server-status  
</Location>
```

You could also use this directive to configure a particular handler for files with a particular file extension. For example:

```
<FilesMatch "\.php$">  
    SetHandler application/x-httpd-php
```

```
</FilesMatch>
```

String-valued expressions can be used to reference per-request variables, including backreferences to named regular expressions:

```
<LocationMatch ^/app/(?<sub>[^/]+)/>  
    SetHandler "proxy:unix:/var/run/app_%{  
</LocationMatch>
```

You can override an earlier defined `SetHandler` directive by using the value `None`.

### Note

Because `SetHandler` overrides default handlers, normal behavior such as handling of URLs ending in a slash (`/`) as directories or index files is suppressed.

### See also

- [AddHandler](#)



<b>Description:</b>	Sets the filters that will process client requests and POST input
<b>Syntax:</b>	<code>SetInputFilter <i>filter</i> [<i>;filter...</i>]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

The `SetInputFilter` directive sets the filter or filters which will process client requests and POST input when they are received by the server. This is in addition to any filters defined elsewhere, including the `AddInputFilter` directive.

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content.

**See also**

- [Filters](#) documentation



<b>Description:</b>	Sets the filters that will process responses from the server
<b>Syntax:</b>	<code>SetOutputFilter <i>filter</i> [<i>;filter...</i>]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

The `SetOutputFilter` directive sets the filters which will process responses from the server before they are sent to the client. This is in addition to any filters defined elsewhere, including the `AddOutputFilter` directive.

For example, the following configuration will process all files in the `/www/data/` directory for server-side includes.

```
<Directory "/www/data/">
  SetOutputFilter INCLUDES
</Directory>
```

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content.

## See also

- [Filters](#) documentation



<b>Description:</b>	Amount of time the server will wait for certain events before failing a request
<b>Syntax:</b>	Timeout <i>seconds</i>
<b>Default:</b>	Timeout 60
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

The **Timeout** directive defines the length of time Apache httpd will wait for I/O in various circumstances:

- When reading data from the client, the length of time to wait for a TCP packet to arrive if the read buffer is empty.

For initial data on a new connection, this directive doesn't take effect until after any configured **AcceptFilter** has passed the new connection to the server.

- When writing data to the client, the length of time to wait for an acknowledgement of a packet if the send buffer is full.
- In **mod\_cgi** and **mod\_cgid**, the length of time to wait for any individual block of output from a CGI script.
- In **mod\_ext\_filter**, the length of time to wait for output from a filtering process.
- In **mod\_proxy**, the default timeout value if **ProxyTimeout** is not configured.



<b>Description:</b>	Determines the behavior on TRACE requests
<b>Syntax:</b>	TraceEnable <i>[on off extended]</i>
<b>Default:</b>	TraceEnable on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

This directive overrides the behavior of TRACE for both the core server and `mod_proxy`. The default `TraceEnable on` permits TRACE requests per RFC 2616, which disallows any request body to accompany the request. `TraceEnable off` causes the core server and `mod_proxy` to return a 405 (Method not allowed) error to the client.

Finally, for testing and diagnostic purposes only, request bodies may be allowed using the non-compliant `TraceEnable extended` directive. The core (as an origin server) will restrict the request body to 64Kb (plus 8Kb for chunk headers if `Transfer-Encoding: chunked` is used). The core will reflect the full headers and all chunk headers with the response body. As a proxy server, the request body is not restricted to 64Kb.

### Note

Despite claims to the contrary, enabling the TRACE method does not expose any security vulnerability in Apache httpd. The TRACE method is defined by the HTTP/1.1 specification and implementations are expected to support it.



<b>Description:</b>	Undefine the existence of a variable
<b>Syntax:</b>	UnDefine <i>parameter-name</i>
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

Undoes the effect of a [Define](#) or of passing a -D argument to [httpd](#).

This directive can be used to toggle the use of [<IfDefine>](#) sections without needing to alter -D arguments in any startup scripts.

While this directive is supported in virtual host context, the changes it makes are visible to any later configuration directives, beyond any enclosing virtual host.

## See also

- [Define](#)
- [IfDefine](#)



**Description:** Configures how the server determines its own name and port

**Syntax:** UseCanonicalName On|Off|DNS

**Default:** UseCanonicalName Off

**Context:** server config, virtual host, directory

**Status:** Core

**Module:** core

In many situations Apache httpd must construct a *self-referential* URL -- that is, a URL that refers back to the same server. With `UseCanonicalName On` Apache httpd will use the hostname and port specified in the `ServerName` directive to construct the canonical name for the server. This name is used in all self-referential URLs, and for the values of `SERVER_NAME` and `SERVER_PORT` in CGIs.

With `UseCanonicalName Off` Apache httpd will form self-referential URLs using the hostname and port supplied by the client if any are supplied (otherwise it will use the canonical name, as defined above). These values are the same that are used to implement [name-based virtual hosts](#) and are available with the same clients. The CGI variables `SERVER_NAME` and `SERVER_PORT` will be constructed from the client supplied values as well.

An example where this may be useful is on an intranet server where you have users connecting to the machine using short names such as `www`. You'll notice that if the users type a shortname and a URL which is a directory, such as `http://www/sp1at`, *without the trailing slash*, then Apache httpd will redirect them to `http://www.example.com/sp1at/`. If you have authentication enabled, this will cause the user to have to

authenticate twice (once for `www` and once again for `www.example.com` -- see [the FAQ on this subject for more information](#)). But if `UseCanonicalName` is set `Off`, then Apache `httpd` will redirect to `http://www/splat/`.

There is a third option, `UseCanonicalName DNS`, which is intended for use with mass IP-based virtual hosting to support ancient clients that do not provide a `Host:` header. With this option, Apache `httpd` does a reverse DNS lookup on the server IP address that the client connected to in order to work out self-referential URLs.

### Warning

If CGIs make assumptions about the values of `SERVER_NAME`, they may be broken by this option. The client is essentially free to give whatever value they want as a hostname. But if the CGI is only using `SERVER_NAME` to construct self-referential URLs, then it should be just fine.

### See also

- [UseCanonicalPhysicalPort](#)
- [ServerName](#)
- [Listen](#)



<b>Description:</b>	Configures how the server determines its own port
<b>Syntax:</b>	UseCanonicalPhysicalPort On Off
<b>Default:</b>	UseCanonicalPhysicalPort Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core

In many situations Apache httpd must construct a *self-referential* URL -- that is, a URL that refers back to the same server. With `UseCanonicalPhysicalPort On`, Apache httpd will, when constructing the canonical port for the server to honor the `UseCanonicalName` directive, provide the actual physical port number being used by this request as a potential port. With `UseCanonicalPhysicalPort Off`, Apache httpd will not ever use the actual physical port number, instead relying on all configured information to construct a valid port number.

### Note

The ordering of the lookup when the physical port is used is as follows:

#### **UseCanonicalName On**

1. Port provided in `Servername`
2. Physical port
3. Default port

#### **UseCanonicalName Off | DNS**

1. Parsed port from Host : header
2. Physical port
3. Port provided in `Servername`

#### 4. Default port

With `UseCanonicalPhysicalPort` Off, the physical ports are removed from the ordering.

#### See also

- [UseCanonicalName](#)
- [ServerName](#)
- [Listen](#)



<b>Description:</b>	Contains directives that apply only to a specific hostname or IP address
<b>Syntax:</b>	<code>&lt;VirtualHost addr[:port] [addr[:port]] ...&gt; ... &lt;/VirtualHost&gt;</code>
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

`<VirtualHost>` and `</VirtualHost>` are used to enclose a group of directives that will apply only to a particular virtual host. Any directive that is allowed in a virtual host context may be used. When the server receives a request for a document on a particular virtual host, it uses the configuration directives enclosed in the `<VirtualHost>` section. *Addr* can be any of the following, optionally followed by a colon and a port number (or \*):

- The IP address of the virtual host;
- A fully qualified domain name for the IP address of the virtual host (not recommended);
- The character \*, which acts as a wildcard and matches any IP address.
- The string `_default_`, which is an alias for \*

```
<VirtualHost 10.1.2.3:80>
  ServerAdmin webmaster@host.example.com
  DocumentRoot "/www/docs/host.example.com"
  ServerName host.example.com
  ErrorLog "logs/host.example.com-error_log"
  TransferLog "logs/host.example.com-access_log"
</VirtualHost>
```

IPv6 addresses must be specified in square brackets because the

optional port number could not be determined otherwise. An IPv6 example is shown below:

```
<VirtualHost [2001:db8::a00:20ff:fea7:ccea]
  ServerAdmin webmaster@host.example.com
  DocumentRoot "/www/docs/host.example.com"
  ServerName host.example.com
  ErrorLog "logs/host.example.com-error_log"
  TransferLog "logs/host.example.com-access_
</VirtualHost>
```

Each Virtual Host must correspond to a different IP address, different port number, or a different host name for the server, in the former case the server machine must be configured to accept IP packets for multiple addresses. (If the machine does not have multiple network interfaces, then this can be accomplished with the `ifconfig alias` command -- if your OS supports it).

#### Note

The use of `<VirtualHost>` does **not** affect what addresses Apache httpd listens on. You may need to ensure that Apache httpd is listening on the correct addresses using [Listen](#).

A [ServerName](#) should be specified inside each `<VirtualHost>` block. If it is absent, the [ServerName](#) from the "main" server configuration will be inherited.

When a request is received, the server first maps it to the best matching `<VirtualHost>` based on the local IP address and port combination only. Non-wildcards have a higher precedence. If no match based on IP and port occurs at all, the "main" server configuration is used.

If multiple virtual hosts contain the best matching IP address and port, the server selects from these virtual hosts the best match based on the requested hostname. If no matching name-based virtual host is found, then the first listed virtual host that matched the IP address will be used. As a consequence, the first listed virtual host for a given IP address and port combination is the default virtual host for that IP and port combination.

### Security

See the [security tips](#) document for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server.

### See also

- [Apache HTTP Server Virtual Host documentation](#)
- [Issues Regarding DNS and Apache HTTP Server](#)
- [Setting which addresses and ports Apache HTTP Server uses](#)
- [How <Directory>, <Location> and <Files> sections work](#) for an explanation of how these different sections are combined when a request is received

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache MPM Common Directives

**Description:** A collection of directives that are implemented by more than one multi-processing module (MPM)

**Status:** MPM



## CoreDumpDirectory Directive

<b>Description:</b>	Directory where Apache HTTP Server attempts to switch before dumping core
<b>Syntax:</b>	CoreDumpDirectory <i>directory</i>
<b>Default:</b>	See usage for the default setting
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a>

This controls the directory to which Apache httpd attempts to switch before dumping core. If your operating system is configured to create core files in the working directory of the crashing process, **CoreDumpDirectory** is necessary to change working directory from the default **ServerRoot** directory, which should not be writable by the user the server runs as.

If you want a core dump for debugging, you can use this directive to place it in a different location. This directive has no effect if your operating system is not configured to write core files to the working directory of the crashing processes.

### Core Dumps on Linux

If Apache httpd starts as root and switches to another user, the Linux kernel *disables* core dumps even if the directory is writable for the process. Apache httpd (2.0.46 and later) reenables core dumps on Linux 2.4 and beyond, but only if you explicitly configure a **CoreDumpDirectory**.

### Core Dumps on BSD

To enable core-dumping of suid-executables on BSD-systems (such as FreeBSD), set `kern.sugid_coredump` to 1.

## Specific signals

`CoreDumpDirectory` processing only occurs for a select set of fatal signals: SIGFPE, SIGILL, SIGABORT, SIGSEGV, and SIGBUS.

On some operating systems, SIGQUIT also results in a core dump but does not go through `CoreDumpDirectory` or `EnableExceptionHook` processing, so the core location is dictated entirely by the operating system.



<b>Description:</b>	Enables a hook that runs exception handlers after a crash
<b>Syntax:</b>	EnableExceptionHook On Off
<b>Default:</b>	EnableExceptionHook Off
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a>

For safety reasons this directive is only available if the server was configured with the `--enable-exception-hook` option. It enables a hook that allows external modules to plug in and do something after a child crashed.

There are already two modules, `mod_whatkilledus` and `mod_backtrace` that make use of this hook. Please have a look at Jeff Trawick's [EnableExceptionHook site](#) for more information about these.



<b>Description:</b>	Specify a timeout after which a gracefully shutdown server will exit.
<b>Syntax:</b>	<code>GracefulShutdownTimeout</code> <i>seconds</i>
<b>Default:</b>	<code>GracefulShutdownTimeout</code> 0
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a>
<b>Compatibility:</b>	Available in version 2.2 and later

The `GracefulShutdownTimeout` specifies how many seconds after receiving a "graceful-stop" signal, a server should continue to run, handling the existing connections.

Setting this value to zero means that the server will wait indefinitely until all remaining requests have been fully served.



<b>Description:</b>	IP addresses and ports that the server listens to
<b>Syntax:</b>	<code>Listen [IP-address:]portnumber [protocol]</code>
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_netware</a> , <a href="#">mpm_os2</a>
<b>Compatibility:</b>	The <i>protocol</i> argument was added in 2.1.5

The `Listen` directive instructs Apache httpd to listen to only specific IP addresses or ports; by default it responds to requests on all IP interfaces. `Listen` is now a required directive. If it is not in the config file, the server will fail to start. This is a change from previous versions of Apache httpd.

The `Listen` directive tells the server to accept incoming requests on the specified port or address-and-port combination. If only a port number is specified, the server listens to the given port on all interfaces. If an IP address is given as well as a port, the server will listen on the given port and interface.

Multiple `Listen` directives may be used to specify a number of addresses and ports to listen to. The server will respond to requests from any of the listed addresses and ports.

For example, to make the server accept connections on both port 80 and port 8000, use:

```
Listen 80  
Listen 8000
```

To make the server accept connections on two specified interfaces and port numbers, use

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6 addresses must be surrounded in square brackets, as in the following example:

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

The optional *protocol* argument is not required for most configurations. If not specified, `https` is the default for port 443 and `http` the default for all other ports. The protocol is used to determine which module should handle a request, and to apply protocol specific optimizations with the [AcceptFilter](#) directive.

You only need to set the protocol if you are running on non-standard ports. For example, running an `https` site on port 8443:

```
Listen 192.170.2.1:8443 https
```

### Error condition

Multiple `Listen` directives for the same ip address and port will result in an `Address already in use` error message.

### See also

- [DNS Issues](#)
- [Setting which addresses and ports Apache HTTP Server uses](#)
- [Further discussion of the Address already in use error message, including other causes.](#)



<b>Description:</b>	Maximum length of the queue of pending connections
<b>Syntax:</b>	ListenBacklog <i>backlog</i>
<b>Default:</b>	ListenBacklog 511
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a>

The maximum length of the queue of pending connections. Generally no tuning is needed or desired, however on some systems it is desirable to increase this when under a TCP SYN flood attack. See the backlog parameter to the `listen(2)` system call.

This will often be limited to a smaller number by the operating system. This varies from OS to OS. Also note that many OSes do not use exactly what is specified as the backlog, but use a number based on (but normally larger than) what is set.



<b>Description:</b>	Ratio between the number of CPU cores (online) and the number of listeners' buckets
<b>Syntax:</b>	ListenCoresBucketsRatio <i>ratio</i>
<b>Default:</b>	ListenCoresBucketsRatio 0 (disabled)
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a>
<b>Compatibility:</b>	Available in Apache HTTP Server 2.4.17, with a kernel supporting the socket option SO_REUSEPORT and distributing new connections evenly across listening processes' (or threads') sockets using it (eg. Linux 3.9 and later, but not the current implementations of SO_REUSEPORT in *BSDs).

A *ratio* between the number of (online) CPU cores and the number of listeners' buckets can be used to make Apache HTTP Server create  $\text{num\_cpu\_cores} / \text{ratio}$  listening buckets, each containing its own **Listen**-ing socket(s) on the same port(s), and then make each child handle a single bucket (with round-robin distribution of the buckets at children creation time).

### Meaning of "online" CPU core

On Linux (and also BSD) a CPU core can be turned on/off if [Hotplug](#) is configured, therefore **ListenCoresBucketsRatio** needs to take this parameter into account while calculating the number of buckets to create.

**ListenCoresBucketsRatio** can improve the scalability when accepting new connections is/becomes the bottleneck. On systems with a large number of CPU cores, enabling this feature

has been tested to show significant performances improvement and shorter responses time.

There must be at least twice the number of CPU cores than the configured *ratio* for this to be active. The recommended *ratio* is 8, hence at least 16 cores should be available at runtime when this value is used. The right *ratio* to obtain maximum performance needs to be calculated for each target system, testing multiple values and observing the variations in your key performance metrics.

This directive influences the calculation of the [MinSpareThreads](#) and [MaxSpareThreads](#) lower bound values. The number of children processes needs to be a multiple of the number of buckets to optimally accept connections.

### **Multiple [Listeners](#) or Apache HTTP servers on the same IP address and port**

Setting the `SO_REUSEPORT` option on the listening socket(s) consequently allows multiple processes (sharing the same EUID, e.g. `root`) to bind to the the same IP address and port, without the binding error raised by the system in the usual case.

This also means that multiple instances of Apache `httpd` configured on a same `IP : port` and with a positive [ListenCoresBucketsRatio](#) would start without an error too, and then run with incoming connections evenly distributed accross both instances (this is NOT a recommendation or a sensible usage in any case, but just a notice that it would prevent such possible issues to be detected).

Within the same instance, Apache `httpd` will check and fail to start if multiple [Listen](#) directives on the exact same IP (or hostname) and port are configured, thus avoiding the creation of some duplicated buckets which would be useless and kill

performances. However it can't (and won't try harder to) catch all the possible overlapping cases (like a hostname resolving to an IP used elsewhere).



## MAXCONNECTIONSPERCHILD DIRECTIVE

<b>Description:</b>	Limit on the number of connections that an individual child server will handle during its life
<b>Syntax:</b>	MaxConnectionsPerChild <i>number</i>
<b>Default:</b>	MaxConnectionsPerChild 0
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_network</a> , <a href="#">mpm_os2</a>
<b>Compatibility:</b>	Available Apache HTTP Server 2.3.9 and later. The old name MaxRequestsPerChild is still supported.

The `MaxConnectionsPerChild` directive sets the limit on the number of connections that an individual child server process will handle. After `MaxConnectionsPerChild` connections, the child process will die. If `MaxConnectionsPerChild` is 0, then the process will never expire.

Setting `MaxConnectionsPerChild` to a non-zero value limits the amount of memory that process can consume by (accidental) memory leakage.



<b>Description:</b>	Maximum amount of memory that the main allocator is allowed to hold without calling <code>free()</code>
<b>Syntax:</b>	<code>MaxMemFree</code> <i>KBytes</i>
<b>Default:</b>	<code>MaxMemFree</code> 2048
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_netware</a>

The `MaxMemFree` directive sets the maximum number of free Kbytes that every allocator is allowed to hold without calling `free()`. In threaded MPMs, every thread has its own allocator. When set to zero, the threshold will be set to unlimited.



<b>Description:</b>	Maximum number of connections that will be processed simultaneously
<b>Syntax:</b>	<code>MaxRequestWorkers</code> <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<code>event</code> , <code>worker</code> , <code>prefork</code>

The `MaxRequestWorkers` directive sets the limit on the number of simultaneous requests that will be served. Any connection attempts over the `MaxRequestWorkers` limit will normally be queued, up to a number based on the `ListenBacklog` directive. Once a child process is freed at the end of a different request, the connection will then be serviced.

For non-threaded servers (*i.e.*, `prefork`), `MaxRequestWorkers` translates into the maximum number of child processes that will be launched to serve requests. The default value is 256; to increase it, you must also raise `ServerLimit`.

For threaded and hybrid servers (e.g. `event` or `worker`) `MaxRequestWorkers` restricts the total number of threads that will be available to serve clients. For hybrid MPMs the default value is 16 (`ServerLimit`) multiplied by the value of 25 (`ThreadsPerChild`). Therefore, to increase `MaxRequestWorkers` to a value that requires more than 16 processes, you must also raise `ServerLimit`.

`MaxRequestWorkers` was called `MaxClients` before version 2.3.13. The old name is still supported.



## MaxSpareThreads Directive

<b>Description:</b>	Maximum number of idle threads
<b>Syntax:</b>	MaxSpareThreads <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a>

Maximum number of idle threads. Different MPMs deal with this directive differently.

For [worker](#) and [event](#), the default is MaxSpareThreads 250. These MPMs deal with idle threads on a server-wide basis. If there are too many idle threads in the server then child processes are killed until the number of idle threads is less than this number. Additional processes/threads might be created if [ListenCoresBucketsRatio](#) is enabled.

For [mpm\\_netware](#) the default is MaxSpareThreads 100. Since this MPM runs a single-process, the spare thread count is also server-wide.

[mpmt\\_os2](#) works similar to [mpm\\_netware](#). For [mpmt\\_os2](#) the default value is 10.

### Restrictions

The range of the [MaxSpareThreads](#) value is restricted. Apache httpd will correct the given value automatically according to the following rules:

- [mpm\\_netware](#) wants the value to be greater than [MinSpareThreads](#).
- For [worker](#) and [event](#), the value must be greater or equal

to the sum of [MinSpareThreads](#) and [ThreadsPerChild](#).

## See also

- [MinSpareThreads](#)
- [StartServers](#)
- [MaxSpareServers](#)



<b>Description:</b>	Minimum number of idle threads available to handle request spikes
<b>Syntax:</b>	MinSpareThreads <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a>

Minimum number of idle threads to handle request spikes. Different MPMs deal with this directive differently.

[worker](#) and [event](#) use a default of MinSpareThreads 75 and deal with idle threads on a server-wide basis. If there aren't enough idle threads in the server then child processes are created until the number of idle threads is greater than *number*. Additional processes/threads might be created if [ListenCoresBucketsRatio](#) is enabled.

[mpm\\_netware](#) uses a default of MinSpareThreads 10 and, since it is a single-process MPM, tracks this on a server-wide bases.

[mpmt\\_os2](#) works similar to [mpm\\_netware](#). For [mpmt\\_os2](#) the default value is 5.

## See also

- [MaxSpareThreads](#)
- [StartServers](#)
- [MinSpareServers](#)



<b>Description:</b>	File where the server records the process ID of the daemon
<b>Syntax:</b>	<code>PidFile filename</code>
<b>Default:</b>	<code>PidFile logs/httpd.pid</code>
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_os2</a>

The `PidFile` directive sets the file to which the server records the process id of the daemon. If the filename is not absolute then it is assumed to be relative to the `ServerRoot`.

### Example

```
PidFile /var/run/apache.pid
```

It is often useful to be able to send the server a signal, so that it closes and then re-opens its `ErrorLog` and `TransferLog`, and re-reads its configuration files. This is done by sending a SIGHUP (kill -1) signal to the process id listed in the `PidFile`.

The `PidFile` is subject to the same warnings about log file placement and [security](#).

### Note

As of Apache HTTP Server 2, we recommended that you only use the [apachectl](#) script, or the init script that your OS provides, for (re-)starting or stopping the server.



<b>Description:</b>	TCP receive buffer size
<b>Syntax:</b>	ReceiveBufferSize <i>bytes</i>
<b>Default:</b>	ReceiveBufferSize 0
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a>

The server will set the TCP receive buffer size to the number of bytes specified.

If set to the value of 0, the server will use the OS default.



<b>Description:</b>	Location of the file used to store coordination data for the child processes
<b>Syntax:</b>	ScoreBoardFile <i>file-path</i>
<b>Default:</b>	ScoreBoardFile logs/apache_runtime_status
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a>

Apache HTTP Server uses a scoreboard to communicate between its parent and child processes. Some architectures require a file to facilitate this communication. If the file is left unspecified, Apache httpd first attempts to create the scoreboard entirely in memory (using anonymous shared memory) and, failing that, will attempt to create the file on disk (using file-based shared memory). Specifying this directive causes Apache httpd to always create the file on the disk.

### Example

```
ScoreBoardFile /var/run/apache_runtime_status
```

File-based shared memory is useful for third-party applications that require direct access to the scoreboard.

If you use a **ScoreBoardFile** then you may see improved speed by placing it on a RAM disk. But be careful that you heed the same warnings about log file placement and [security](#).

### See also

- [Stopping and Restarting Apache HTTP Server](#)



<b>Description:</b>	TCP buffer size
<b>Syntax:</b>	SendBufferSize <i>bytes</i>
<b>Default:</b>	SendBufferSize 0
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a>

Sets the server's TCP send buffer size to the number of bytes specified. It is often useful to set this past the OS's standard default value on high speed, high latency connections (*i.e.*, 100ms or so, such as transcontinental fast pipes).

If set to the value of 0, the server will use the default value provided by your OS.

Further configuration of your operating system may be required to elicit better performance on high speed, high latency connections.

On some operating systems, changes in TCP behavior resulting from a larger `SendBufferSize` may not be seen unless `EnableSendfile` is set to OFF. This interaction applies only to static files.



<b>Description:</b>	Upper limit on configurable number of processes
<b>Syntax:</b>	ServerLimit <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a>

For the [prefork](#) MPM, this directive sets the maximum configured value for [MaxRequestWorkers](#) for the lifetime of the Apache httpd process. For the [worker](#) and [event](#) MPMs, this directive in combination with [ThreadLimit](#) sets the maximum configured value for [MaxRequestWorkers](#) for the lifetime of the Apache httpd process. For the [event](#) MPM, this directive also defines how many old server processes may keep running and finish processing open connections. Any attempts to change this directive during a restart will be ignored, but [MaxRequestWorkers](#) can be modified during a restart.

Special care must be taken when using this directive. If [ServerLimit](#) is set to a value much higher than necessary, extra, unused shared memory will be allocated. If both [ServerLimit](#) and [MaxRequestWorkers](#) are set to values higher than the system can handle, Apache httpd may not start or the system may become unstable.

With the [prefork](#) MPM, use this directive only if you need to set [MaxRequestWorkers](#) higher than 256 (default). Do not set the value of this directive any higher than what you might want to set [MaxRequestWorkers](#) to.

With [worker](#), use this directive only if your [MaxRequestWorkers](#) and [ThreadsPerChild](#) settings require more than 16 server

processes (default). Do not set the value of this directive any higher than the number of server processes required by what you may want for [MaxRequestWorkers](#) and [ThreadsPerChild](#).

With [event](#), increase this directive if the process number defined by your [MaxRequestWorkers](#) and [ThreadsPerChild](#) settings, plus the number of gracefully shutting down processes, is more than 16 server processes (default).

### Note

There is a hard limit of `ServerLimit 20000` compiled into the server (for the [prefork](#) MPM 200000). This is intended to avoid nasty effects caused by typos. To increase it even further past this limit, you will need to modify the value of `MAX_SERVER_LIMIT` in the mpm source file and rebuild the server.

### See also

- [Stopping and Restarting Apache HTTP Server](#)



<b>Description:</b>	Number of child server processes created at startup
<b>Syntax:</b>	StartServers <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">prefork</a> , <a href="#">mpmt_os2</a>

The `StartServers` directive sets the number of child server processes created on startup. As the number of processes is dynamically controlled depending on the load, (see [MinSpareThreads](#), [MaxSpareThreads](#), [MinSpareServers](#), [MaxSpareServers](#)) there is usually little reason to adjust this parameter.

The default value differs from MPM to MPM. [worker](#) and [event](#) default to `StartServers 3`; [prefork](#) defaults to 5; [mpmt\\_os2](#) defaults to 2.



<b>Description:</b>	Number of threads created on startup
<b>Syntax:</b>	StartThreads <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">mpm_netware</a>

Number of threads created on startup. As the number of threads is dynamically controlled depending on the load, (see [MinSpareThreads](#), [MaxSpareThreads](#), [MinSpareServers](#), [MaxSpareServers](#)) there is usually little reason to adjust this parameter.

For [mpm\\_netware](#) the default is StartThreads 50 and, since there is only a single process, this is the total number of threads created at startup to serve requests.



<b>Description:</b>	Sets the upper limit on the configurable number of threads per child process
<b>Syntax:</b>	ThreadLimit <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">mpm_winnt</a>

This directive sets the maximum configured value for [ThreadsPerChild](#) for the lifetime of the Apache httpd process. Any attempts to change this directive during a restart will be ignored, but [ThreadsPerChild](#) can be modified during a restart up to the value of this directive.

Special care must be taken when using this directive. If [ThreadLimit](#) is set to a value much higher than [ThreadsPerChild](#), extra unused shared memory will be allocated. If both [ThreadLimit](#) and [ThreadsPerChild](#) are set to values higher than the system can handle, Apache httpd may not start or the system may become unstable. Do not set the value of this directive any higher than your greatest predicted setting of [ThreadsPerChild](#) for the current run of Apache httpd.

The default value for [ThreadLimit](#) is 1920 when used with [mpm\\_winnt](#) and 64 when used with the others.

### Note

There is a hard limit of `ThreadLimit 20000` (or `ThreadLimit 100000` with [event](#), `ThreadLimit 15000` with [mpm\\_winnt](#)) compiled into the server. This is intended to avoid nasty effects caused by typos. To increase it even further past this limit, you will need to modify the value of

MAX\_THREAD\_LIMIT in the mpm source file and rebuild the server.



## ThreadsPerChild Directive

<b>Description:</b>	Number of threads created by each child process
<b>Syntax:</b>	ThreadsPerChild <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">mpm_winnt</a>

This directive sets the number of threads created by each child process. The child creates these threads at startup and never creates more. If using an MPM like [mpm\\_winnt](#), where there is only one child process, this number should be high enough to handle the entire load of the server. If using an MPM like [worker](#), where there are multiple child processes, the *total* number of threads should be high enough to handle the common load on the server.

The default value for **ThreadsPerChild** is 64 when used with [mpm\\_winnt](#) and 25 when used with the others.



<b>Description:</b>	The size in bytes of the stack used by threads handling client connections
<b>Syntax:</b>	<code>ThreadStackSize size</code>
<b>Default:</b>	65536 on NetWare; varies on other operating systems
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">worker</a> , <a href="#">mpm_winnt</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a>
<b>Compatibility:</b>	Available in Apache HTTP Server 2.1 and later

The `ThreadStackSize` directive sets the size of the stack (for autodata) of threads which handle client connections and call modules to help process those connections. In most cases the operating system default for stack size is reasonable, but there are some conditions where it may need to be adjusted:

- On platforms with a relatively small default thread stack size (e.g., HP-UX), Apache httpd may crash when using some third-party modules which use a relatively large amount of autodata storage. Those same modules may have worked fine on other platforms where the default thread stack size is larger. This type of crash is resolved by setting `ThreadStackSize` to a value higher than the operating system default. This type of adjustment is necessary only if the provider of the third-party module specifies that it is required, or if diagnosis of an Apache httpd crash indicates that the thread stack size was too small.
- On platforms where the default thread stack size is significantly larger than necessary for the web server configuration, a higher number of threads per child process will be achievable if `ThreadStackSize` is set to a value

lower than the operating system default. This type of adjustment should only be made in a test environment which allows the full set of web server processing can be exercised, as there may be infrequent requests which require more stack to process. The minimum required stack size strongly depends on the modules used, but any change in the web server configuration can invalidate the current `ThreadStackSize` setting.

- On Linux, this directive can only be used to increase the default stack size, as the underlying system call uses the value as a *minimum* stack size. The (often large) soft limit for `ulimit -s` (8MB if unlimited) is used as the default stack size.

It is recommended to not reduce `ThreadStackSize` unless a high number of threads per child process is needed. On some platforms (including Linux), a setting of 128000 is already too low and causes crashes with some common modules.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache MPM event

<b>Description:</b>	A variant of the <a href="#">worker</a> MPM with the goal of consuming threads only for connections with active processing
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_event_module
<b>Source File:</b>	event.c

## Summary

The [event](#) Multi-Processing Module (MPM) is designed to allow more requests to be served simultaneously by passing off some processing work to the listeners threads, freeing up the worker threads to serve new requests.

To use the [event](#) MPM, add `--with-mpm=event` to the [configure](#) script's arguments when building the [httpd](#).



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[The worker MPM](#)



**Relationship with the Worker MPM**

`event` is based on the `worker` MPM, which implements a hybrid multi-process multi-threaded server. A single control process (the parent) is responsible for launching child processes. Each child process creates a fixed number of server threads as specified in the `ThreadsPerChild` directive, as well as a listener thread which listens for connections and passes them to a worker thread for processing when they arrive.

Run-time configuration directives are identical to those provided by `worker`, with the only addition of the `AsyncRequestWorkerFactor`.



This MPM tries to fix the 'keep alive problem' in HTTP. After a client completes the first request, it can keep the connection open, sending further requests using the same socket and saving significant overhead in creating TCP connections. However, Apache HTTP Server traditionally keeps an entire child process/thread waiting for data from the client, which brings its own disadvantages. To solve this problem, this MPM uses a dedicated listener thread for each process to handle both the Listening sockets, all sockets that are in a Keep Alive state, sockets where the handler and protocol filters have done their work and the ones where the only remaining thing to do is send the data to the client.

This new architecture, leveraging non-blocking sockets and modern kernel features exposed by [APR](#) (like Linux's `epoll`), no longer requires the `mpm-accept` [Mutex](#) configured to avoid the thundering herd problem.

The total amount of connections that a single process/threads block can handle is regulated by the [AsyncRequestWorkerFactor](#) directive.

## Async connections

Async connections would need a fixed dedicated worker thread with the previous MPMs but not with `event`. The status page of [mod\\_status](#) shows new columns under the Async connections section:

### Writing

While sending the response to the client, it might happen that the TCP write buffer fills up because the connection is too slow. Usually in this case a `write()` to the socket returns `EWOULDBLOCK` or `EAGAIN`, to become writable again after an

idle time. The worker holding the socket might be able to offload the waiting task to the listener thread, that in turn will re-assign it to the first idle worker thread available once an event will be raised for the socket (for example, "the socket is now writable"). Please check the Limitations section for more information.

### **Keep-alive**

Keep Alive handling is the most basic improvement from the worker MPM. Once a worker thread finishes to flush the response to the client, it can offload the socket handling to the listener thread, that in turns will wait for any event from the OS, like "the socket is readable". If any new request comes from the client, then the listener will forward it to the first worker thread available. Conversely, if the [KeepAliveTimeout](#) occurs then the socket will be closed by the listener. In this way the worker threads are not responsible for idle sockets and they can be re-used to serve other requests.

### **Closing**

Sometimes the MPM needs to perform a lingering close, namely sending back an early error to the client while it is still transmitting data to httpd. Sending the response and then closing the connection immediately is not the correct thing to do since the client (still trying to send the rest of the request) would get a connection reset and could not read the httpd's response. So in such cases, httpd tries to read the rest of the request to allow the client to consume the response. The lingering close is time bounded but it can take relatively long time, so a worker thread can offload this work to the listener.

These improvements are valid for both HTTP/HTTPS connections.

## **Graceful process termination and Scoreboard usage**

This mpm showed some scalability bottlenecks in the past leading to the following error: "**scoreboard is full, not at MaxRequestWorkers**". [MaxRequestWorkers](#) limits the number of simultaneous requests that will be served at any given time and also the number of allowed processes ([MaxRequestWorkers / ThreadsPerChild](#)), meanwhile the Scoreboard is a representation of all the running processes and the status of their worker threads. If the scoreboard is full (so all the threads have a state that is not idle) but the number of active requests served is not [MaxRequestWorkers](#), it means that some of them are blocking new requests that could be served but that are queued instead (up to the limit imposed by [ListenBacklog](#)). Most of the times the threads are stuck in the Graceful state, namely they are waiting to finish their work with a TCP connection to safely terminate and free up a scoreboard slot (for example handling long running requests, slow clients or connections with keep-alive enabled). Two scenarios are very common:

- During a [graceful restart](#). The parent process signals all its children to complete their work and terminate, while it reloads the config and forks new processes. If the old children keep running for a while before stopping, the scoreboard will be partially occupied until their slots are freed.
- When the server load goes down in a way that causes httpd to stop some processes (for example due to [MaxSpareThreads](#)). This is particularly problematic because when the load increases again, httpd will try to start new processes. If the pattern repeats, the number of processes can rise quite a bit, ending up in a mixture of old processes trying to stop and new ones trying to do some work.

From 2.4.24 onward, mpm-event is smarter and it is able to handle graceful terminations in a much better way. Some of the improvements are:

- Allow the use of all the scoreboard slots up to `ServerLimit`. `MaxRequestWorkers` and `ThreadsPerChild` are used to limit the amount of active processes, meanwhile `ServerLimit` takes also into account the ones doing a graceful close to allow extra slots when needed. The idea is to use `ServerLimit` to instruct httpd about how many overall processes are tolerated before impacting the system resources.
- Force gracefully finishing processes to close their connections in keep-alive state.
- During graceful shutdown, if there are more running worker threads than open connections for a given process, terminate these threads to free resources faster (which may be needed for new processes).
- If the scoreboard is full, prevent more processes to finish gracefully due to reduced load until old processes have terminated (otherwise the situation would get worse once the load increases again).

The behavior described in the last point is completely observable via `mod_status` in the connection summary table through two new columns: "Slot" and "Stopping". The former indicates the PID and the latter if the process is stopping or not; the extra state "Yes (old gen)" indicates a process still running after a graceful restart.

## Limitations

The improved connection handling may not work for certain connection filters that have declared themselves as incompatible with event. In these cases, this MPM will fall back to the behavior of the `worker` MPM and reserve one worker thread per connection. All modules shipped with the server are compatible with the event MPM.

A similar restriction is currently present for requests involving an

output filter that needs to read and/or modify the whole response body. If the connection to the client blocks while the filter is processing the data, and the amount of data produced by the filter is too big to be buffered in memory, the thread used for the request is not freed while httpd waits until the pending data is sent to the client.

To illustrate this point we can think about the following two situations: serving a static asset (like a CSS file) versus serving content retrieved from FCGI/CGI or a proxied server. The former is predictable, namely the event MPM has full visibility on the end of the content and it can use events: the worker thread serving the response content can flush the first bytes until EWOULDBLOCK or EAGAIN is returned, delegating the rest to the listener. This one in turn waits for an event on the socket, and delegates the work to flush the rest of the content to the first idle worker thread.

Meanwhile in the latter example (FCGI/CGI/proxied content) the MPM can't predict the end of the response and a worker thread has to finish its work before returning the control to the listener. The only alternative is to buffer the response in memory, but it wouldn't be the safest option for the sake of the server's stability and memory footprint.

## **Background material**

The event model was made possible by the introduction of new APIs into the supported operating systems:

- `epoll` (Linux)
- `kqueue` (BSD)
- `event ports` (Solaris)

Before these new APIs were made available, the traditional `select` and `poll` APIs had to be used. Those APIs get slow if used to handle many connections or if the set of connections rate of change is high. The new APIs allow to monitor much more

connections and they perform way better when the set of connections to monitor changes frequently. So these APIs made it possible to write the event MPM, that scales much better with the typical HTTP pattern of many idle connections.

The MPM assumes that the underlying `apr_pollset` implementation is reasonably threadsafe. This enables the MPM to avoid excessive high level locking, or having to wake up the listener thread in order to send it a keep-alive socket. This is currently only compatible with KQueue and EPoll.



## Requirements

This MPM depends on [APR](#)'s atomic compare-and-swap operations for thread synchronization. If you are compiling for an x86 target and you don't need to support 386s, or you are compiling for a SPARC and you don't need to run on pre-UltraSPARC chips, add `--enable-nonportable-atomics=yes` to the [configure](#) script's arguments. This will cause APR to implement atomic operations using efficient opcodes not available in older CPUs.

This MPM does not perform well on older platforms which lack good threading, but the requirement for EPoll or KQueue makes this moot.

- To use this MPM on FreeBSD, FreeBSD 5.3 or higher is recommended. However, it is possible to run this MPM on FreeBSD 5.2.1, if you use `libkse` (see `man libmap.conf`).
- For NetBSD, at least version 2.0 is recommended.
- For Linux, a 2.6 kernel is recommended. It is also necessary to ensure that your version of `glibc` has been compiled with support for EPoll.



<b>Description:</b>	Limit concurrent connections per process
<b>Syntax:</b>	AsyncRequestWorkerFactor <i>factor</i>
<b>Default:</b>	2
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	event
<b>Compatibility:</b>	Available in version 2.3.13 and later

The event MPM handles some connections in an asynchronous way, where request worker threads are only allocated for short periods of time as needed, and other connections with one request worker thread reserved per connection. This can lead to situations where all workers are tied up and no worker thread is available to handle new work on established async connections.

To mitigate this problem, the event MPM does two things:

- it limits the number of connections accepted per process, depending on the number of idle request workers;
- if all workers are busy, it will close connections in keep-alive state even if the keep-alive timeout has not expired. This allows the respective clients to reconnect to a different process which may still have worker threads available.

This directive can be used to fine-tune the per-process connection limit. A **process** will only accept new connections if the current number of connections (not counting connections in the "closing" state) is lower than:

$$\text{ThreadsPerChild} + (\text{AsyncRequestWorkerFactor} * \text{number of idle workers})$$

An estimation of the maximum concurrent connections across all

the processes given an average value of idle worker threads can be calculated with:

$$(\text{ThreadsPerChild} + (\text{AsyncRequestWorkerFactor} * \text{number of idle workers})) * \text{ServerLimit}$$

### Example

```
ThreadsPerChild = 10
ServerLimit = 4
AsyncRequestWorkerFactor = 2
MaxRequestWorkers = 40

idle_workers = 4 (average for all the processes to keep it
max_connections = (ThreadsPerChild + (AsyncRequestWorkerFactor *
                    = (10 + (2 * 4)) * 4 = 72
```

When all the worker threads are idle, then absolute maximum numbers of concurrent connections can be calculated in a simpler way:

$$(\text{AsyncRequestWorkerFactor} + 1) * \text{MaxRequestWorkers}$$

### Example

```
ThreadsPerChild = 10
ServerLimit = 4
MaxRequestWorkers = 40
AsyncRequestWorkerFactor = 2
```

If all the processes have all threads idle then:

```
idle_workers = 10
```

We can calculate the absolute maximum numbers of concurrent connections in two ways:

```
max_connections = (ThreadsPerChild + (AsyncRequestWorkerFactor * MaxRequestWorkers)) * 4 = 120  
max_connections = (AsyncRequestWorkerFactor + 1) * MaxRequestWorkers = (2 + 1) * 40 = 120
```

Tuning `AsyncRequestWorkerFactor` requires knowledge about the traffic handled by httpd in each specific use case, so changing the default value requires extensive testing and data gathering from `mod_status`.

`MaxRequestWorkers` was called `MaxClients` prior to version 2.3.13. The above value shows that the old name did not accurately describe its meaning for the event MPM.

`AsyncRequestWorkerFactor` can take non-integer arguments, e.g "1.5".

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache MPM netware

<b>Description:</b>	Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_netware_module
<b>Source File:</b>	mpm_netware.c

### Summary

This Multi-Processing Module (MPM) implements an exclusively threaded web server that has been optimized for Novell NetWare.

The main thread is responsible for launching child worker threads which listen for connections and serve them when they arrive. Apache HTTP Server always tries to maintain several *spare* or idle worker threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new child threads to be spawned before their requests can be served.

The [StartThreads](#), [MinSpareThreads](#), [MaxSpareThreads](#), and [MaxThreads](#) regulate how the main thread creates worker threads to serve requests. In general, Apache httpd is very self-regulating, so most sites do not need to adjust these directives from their default values. Sites with limited memory may need to decrease [MaxThreads](#) to keep the server from thrashing (spawning and terminating idle threads). More information about tuning process creation is provided in the [performance hints](#) documentation.

[MaxConnectionsPerChild](#) controls how frequently the server recycles processes by killing old ones and launching new ones. On the NetWare OS it is highly recommended that this directive remain set to 0. This allows worker threads to continue servicing requests

indefinitely.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Setting which addresses and ports Apache httpd uses](#)



## MaxThreads Directive

<b>Description:</b>	Set the maximum number of worker threads
<b>Syntax:</b>	MaxThreads <i>number</i>
<b>Default:</b>	MaxThreads 2048
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	mpm_netware

The **MaxThreads** directive sets the desired maximum number worker threads allowable. The default value is also the compiled in hard limit. Therefore it can only be lowered, for example:

```
MaxThreads 512
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache MPM os2

<b>Description:</b>	Hybrid multi-process, multi-threaded MPM for OS/2
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_mpmt_os2_module
<b>Source File:</b>	mpmt_os2.c

## Summary

The Server consists of a main, parent process and a small, static number of child processes.

The parent process' job is to manage the child processes. This involves spawning children as required to ensure there are always [StartServers](#) processes accepting connections.

Each child process consists of a pool of worker threads and a main thread that accepts connections and passes them to the workers via a work queue. The worker thread pool is dynamic, managed by a maintenance thread so that the number of idle threads is kept between [MinSpareThreads](#) and [MaxSpareThreads](#).



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

## [Setting which addresses and ports Apache uses](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache MPM prefork

<b>Description:</b>	Implements a non-threaded, pre-forking web server
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_prefork_module
<b>Source File:</b>	prefork.c

## Summary

This Multi-Processing Module (MPM) implements a non-threaded, pre-forking web server. Each server process may answer incoming requests, and a parent process manages the size of the server pool. It is appropriate for sites that need to avoid threading for compatibility with non-thread-safe libraries. It is also the best MPM for isolating each request, so that a problem with a single request will not affect any other.

This MPM is very self-regulating, so it is rarely necessary to adjust its configuration directives. Most important is that [MaxRequestWorkers](#) be big enough to handle as many simultaneous requests as you expect to receive, but small enough to assure that there is enough physical RAM for all processes.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Setting which addresses and ports Apache HTTP Server uses](#)



A single control process is responsible for launching child processes which listen for connections and serve them when they arrive. Apache httpd always tries to maintain several *spare* or idle server processes, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new child processes to be forked before their requests can be served.

The [StartServers](#), [MinSpareServers](#), [MaxSpareServers](#), and [MaxRequestWorkers](#) regulate how the parent process creates children to serve requests. In general, Apache httpd is very self-regulating, so most sites do not need to adjust these directives from their default values. Sites which need to serve more than 256 simultaneous requests may need to increase [MaxRequestWorkers](#), while sites with limited memory may need to decrease [MaxRequestWorkers](#) to keep the server from thrashing (swapping memory to disk and back). More information about tuning process creation is provided in the [performance hints](#) documentation.

While the parent process is usually started as root under Unix in order to bind to port 80, the child processes are launched by Apache httpd as a less-privileged user. The [User](#) and [Group](#) directives are used to set the privileges of the Apache httpd child processes. The child processes must be able to read all the content that will be served, but should have as few privileges beyond that as possible.

[MaxConnectionsPerChild](#) controls how frequently the server recycles processes by killing old ones and launching new ones.

This MPM uses the `mpm-accept` mutex to serialize access to incoming connections when subject to the thundering herd problem (generally, when there are multiple listening sockets). The implementation aspects of this mutex can be configured with the

[Mutex](#) directive. The [performance hints](#) documentation has additional information about this mutex.



<b>Description:</b>	Maximum number of idle child server processes
<b>Syntax:</b>	MaxSpareServers <i>number</i>
<b>Default:</b>	MaxSpareServers 10
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	prefork

The `MaxSpareServers` directive sets the desired maximum number of *idle* child server processes. An idle process is one which is not handling a request. If there are more than `MaxSpareServers` idle, then the parent process will kill off the excess processes.

Tuning of this parameter should only be necessary on very busy sites. Setting this parameter to a large number is almost always a bad idea. If you are trying to set the value equal to or lower than `MinSpareServers`, Apache HTTP Server will automatically adjust it to `MinSpareServers + 1`.

## See also

- [MinSpareServers](#)
- [StartServers](#)
- [MaxSpareThreads](#)



<b>Description:</b>	Minimum number of idle child server processes
<b>Syntax:</b>	<code>MinSpareServers</code> <i>number</i>
<b>Default:</b>	<code>MinSpareServers</code> 5
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	prefork

The `MinSpareServers` directive sets the desired minimum number of *idle* child server processes. An idle process is one which is not handling a request. If there are fewer than `MinSpareServers` idle, then the parent process creates new children: It will spawn one, wait a second, then spawn two, wait a second, then spawn four, and it will continue exponentially until it is spawning 32 children per second. It will stop whenever it satisfies the `MinSpareServers` setting.

Tuning of this parameter should only be necessary on very busy sites. Setting this parameter to a large number is almost always a bad idea.

## See also

- [MaxSpareServers](#)
- [StartServers](#)
- [MinSpareThreads](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache MPM winnt

<b>Description:</b>	Multi-Processing Module optimized for Windows NT.
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_winnt_module
<b>Source File:</b>	mpm_winnt.c

## Summary

This Multi-Processing Module (MPM) is the default for the Windows NT operating systems. It uses a single control process which launches a single child process which in turn creates threads to handle requests

Capacity is configured using the [ThreadsPerChild](#) directive, which sets the maximum number of concurrent client connections.

By default, this MPM uses advanced Windows APIs for accepting new client connections. In some configurations, third-party products may interfere with this implementation, with the following messages written to the web server log:

```
Child: Encountered too many AcceptEx faults accepting client connections.  
winnt_mpm: falling back to 'AcceptFilter none'.
```

The MPM falls back to a safer implementation, but some client requests were not processed correctly. In order to avoid this error, use [AcceptFilter](#) with accept filter none.

```
AcceptFilter http none  
AcceptFilter https none
```

*In Apache httpd 2.0 and 2.2, [Win32DisableAcceptEx](#) was used for this purpose.*

The WinNT MPM differs from the Unix MPMs such as worker and event in several areas:

- When a child process is exiting due to shutdown, restart, or [MaxConnectionsPerChild](#), active requests in the exiting process have [TimeOut](#) seconds to finish before processing is aborted. Alternate types of restart and shutdown are not implemented.
- New child processes read the configuration files instead of inheriting the configuration from the parent. The behavior will be the same as on Unix if the child process is created at startup or restart, but if a child process is created because the prior one crashed or reached [MaxConnectionsPerChild](#), any pending changes to the configuration will become active in the child at that point, and the parent and child will be using a different configuration. If planned configuration changes have been partially implemented and the current configuration cannot be parsed, the replacement child process cannot start up and the server will halt. Because of this behavior, configuration files should not be changed until the time of a server restart.
- The monitor and fatal\_exception hooks are not currently implemented.
- [AcceptFilter](#) is implemented in the MPM and has a different type of control over handling of new connections. (Refer to the [AcceptFilter](#) documentation for details.)



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Using Apache HTTP Server on Microsoft Windows](#)

---

Copyright 2017 The Apache Software Foundation.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache MPM worker

<b>Description:</b>	Multi-Processing Module implementing a hybrid multi-threaded multi-process web server
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_worker_module
<b>Source File:</b>	worker.c

## Summary

This Multi-Processing Module (MPM) implements a hybrid multi-process multi-threaded server. By using threads to serve requests, it is able to serve a large number of requests with fewer system resources than a process-based server. However, it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.

The most important directives used to control this MPM are [ThreadsPerChild](#), which controls the number of threads deployed by each child process and [MaxRequestWorkers](#), which controls the maximum total number of threads that may be launched.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

## Setting which addresses and ports Apache HTTP Server uses



## CHILD PROCESSES

A single control process (the parent) is responsible for launching child processes. Each child process creates a fixed number of server threads as specified in the [ThreadsPerChild](#) directive, as well as a listener thread which listens for connections and passes them to a server thread for processing when they arrive.

Apache HTTP Server always tries to maintain a pool of *spare* or idle server threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new threads or processes to be created before their requests can be served. The number of processes that will initially launch is set by the [StartServers](#) directive. During operation, the server assesses the total number of idle threads in all processes, and forks or kills processes to keep this number within the boundaries specified by [MinSpareThreads](#) and [MaxSpareThreads](#). Since this process is very self-regulating, it is rarely necessary to modify these directives from their default values. The maximum number of clients that may be served simultaneously (i.e., the maximum total number of threads in all processes) is determined by the [MaxRequestWorkers](#) directive. The maximum number of active child processes is determined by the [MaxRequestWorkers](#) directive divided by the [ThreadsPerChild](#) directive.

Two directives set hard limits on the number of active child processes and the number of server threads in a child process, and can only be changed by fully stopping the server and then starting it again. [ServerLimit](#) is a hard limit on the number of active child processes, and must be greater than or equal to the [MaxRequestWorkers](#) directive divided by the [ThreadsPerChild](#) directive. [ThreadLimit](#) is a hard limit of the number of server threads, and must be greater than or equal to the [ThreadsPerChild](#) directive.

In addition to the set of active child processes, there may be additional child processes which are terminating, but where at least one server thread is still handling an existing client connection. Up to [MaxRequestWorkers](#) terminating processes may be present, though the actual number can be expected to be much smaller. This behavior can be avoided by disabling the termination of individual child processes, which is achieved using the following:

- set the value of [MaxConnectionsPerChild](#) to zero
- set the value of [MaxSpareThreads](#) to the same value as [MaxRequestWorkers](#)

A typical configuration of the process-thread controls in the [worker](#) MPM could look as follows:

```
ServerLimit      16
StartServers     2
MaxRequestWorkers 150
MinSpareThreads  25
MaxSpareThreads  75
ThreadsPerChild  25
```

While the parent process is usually started as root under Unix in order to bind to port 80, the child processes and threads are launched by the server as a less-privileged user. The [User](#) and [Group](#) directives are used to set the privileges of the Apache HTTP Server child processes. The child processes must be able to read all the content that will be served, but should have as few privileges beyond that as possible. In addition, unless [suexec](#) is used, these directives also set the privileges which will be inherited by CGI scripts.

[MaxConnectionsPerChild](#) controls how frequently the server

recycles processes by killing old ones and launching new ones.

This MPM uses the `mpm-accept_mutex` to serialize access to incoming connections when subject to the thundering herd problem (generally, when there are multiple listening sockets). The implementation aspects of this mutex can be configured with the [Mutex](#) directive. The [performance hints](#) documentation has additional information about this mutex.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_access_compat`

<b>Description:</b>	Group authorizations based on host (name or IP address)
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>access_compat_module</code>
<b>Source File:</b>	<code>mod_access_compat.c</code>
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3 as a compatibility module with previous versions of Apache httpd 2.x. The directives provided by this module have been deprecated by the new authz refactoring. Please see <a href="#">mod_authz_host</a>

## Summary

The directives provided by [mod\\_access\\_compat](#) are used in `<Directory>`, `<Files>`, and `<Location>` sections as well as `.htaccess` files to control access to particular parts of the server. Access can be controlled based on the client hostname, IP address, or other characteristics of the client request, as captured in [environment variables](#). The `Allow` and `Deny` directives are used to specify which clients are or are not allowed access to the server, while the `Order` directive sets the default access state, and configures how the `Allow` and `Deny` directives interact with each other.

Both host-based access restrictions and password-based authentication may be implemented simultaneously. In that case, the `Satisfy` directive is used to determine how the two sets of restrictions interact.

### Note

The directives provided by [mod\\_access\\_compat](#) have been

deprecated by [mod\\_authz\\_host](#). Mixing old directives like [Order](#), [Allow](#) or [Deny](#) with new ones like [Require](#) is technically possible but discouraged. This module was created to support configurations containing only old directives to facilitate the 2.4 upgrade. Please check the [upgrading](#) guide for more information.

In general, access restriction directives apply to all access methods (GET, PUT, POST, etc). This is the desired behavior in most cases. However, it is possible to restrict some methods, while leaving other methods unrestricted, by enclosing the directives in a [<Limit>](#) section.

### Merging of configuration sections

When any directive provided by this module is used in a new configuration section, no directives provided by this module are inherited from previous configuration sections.



### Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

### See also

[Require](#)

[mod\\_authz\\_host](#)

[mod\\_authz\\_core](#)



**Description:** Controls which hosts can access an area of the server

**Syntax:** Allow from all|*host*|env=[!]env-variable [*host*|env=[!]env-variable]  
...

**Context:** directory, .htaccess

**Override:** Limit

**Status:** Extension

**Module:** mod\_access\_compat

The **Allow** directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, IP address range, or by other characteristics of the client request captured in environment variables.

The first argument to this directive is always `from`. The subsequent arguments can take three different forms. If `Allow from all` is specified, then all hosts are allowed access, subject to the configuration of the **Deny** and **Order** directives as discussed below. To allow only particular hosts or groups of hosts to access the server, the *host* can be specified in any of the following formats:

### A (partial) domain-name

```
Allow from example.org  
Allow from .net example.edu
```

Hosts whose names match, or end in, this string are allowed access. Only complete components are matched, so the above example will match `foo.example.org` but it will not match `fooexample.org`. This configuration will cause

Apache httpd to perform a double DNS lookup on the client IP address, regardless of the setting of the [HostnameLookups](#) directive. It will do a reverse DNS lookup on the IP address to find the associated hostname, and then do a forward lookup on the hostname to assure that it matches the original IP address. Only if the forward and reverse DNS are consistent and the hostname matches will access be allowed.

### **A full IP address**

```
Allow from 10.1.2.3  
Allow from 192.168.1.104 192.168.1.205
```

An IP address of a host allowed access

### **A partial IP address**

```
Allow from 10.1  
Allow from 10 172.20 192.168.2
```

The first 1 to 3 bytes of an IP address, for subnet restriction.

### **A network/netmask pair**

```
Allow from 10.1.0.0/255.255.0.0
```

A network a.b.c.d, and a netmask w.x.y.z. For more fine-grained subnet restriction.

### **A network/nnn CIDR specification**

```
Allow from 10.1.0.0/16
```

Similar to the previous case, except the netmask consists of

nnn high-order 1 bits.

Note that the last three examples above match exactly the same set of hosts.

IPv6 addresses and IPv6 subnets can be specified as shown below:

```
Allow from 2001:db8::a00:20ff:fea7:ccea  
Allow from 2001:db8::a00:20ff:fea7:ccea/10
```

The third format of the arguments to the `Allow` directive allows access to the server to be controlled based on the existence of an [environment variable](#). When `Allow from env=env-variable` is specified, then the request is allowed access if the environment variable `env-variable` exists. When `Allow from env=!env-variable` is specified, then the request is allowed access if the environment variable `env-variable` doesn't exist. The server provides the ability to set environment variables in a flexible way based on characteristics of the client request using the directives provided by `mod_setenvif`. Therefore, this directive can be used to allow access based on such factors as the clients User-Agent (browser type), Referer, or other HTTP request header fields.

```
SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in  
<Directory "/docroot">  
    Order Deny,Allow  
    Deny from all  
    Allow from env=let_me_in  
</Directory>
```

In this case, browsers with a user-agent string beginning with `KnockKnock/2.0` will be allowed access, and all others will be

denied.

### **Merging of configuration sections**

When any directive provided by this module is used in a new configuration section, no directives provided by this module are inherited from previous configuration sections.



**Description:** Controls which hosts are denied access to the server

**Syntax:** Deny from *all|host|env=[!]env-variable [host|env=[!]env-variable]*  
...

**Context:** directory, .htaccess

**Override:** Limit

**Status:** Extension

**Module:** mod\_access\_compat

This directive allows access to the server to be restricted based on hostname, IP address, or environment variables. The arguments for the **Deny** directive are identical to the arguments for the **Allow** directive.



**Description:** Controls the default access state and the order in which **Allow** and **Deny** are evaluated.

**Syntax:** Order *ordering*

**Default:** Order Deny, Allow

**Context:** directory, .htaccess

**Override:** Limit

**Status:** Extension

**Module:** mod\_access\_compat

The **Order** directive, along with the **Allow** and **Deny** directives, controls a three-pass access control system. The first pass processes either all **Allow** or all **Deny** directives, as specified by the **Order** directive. The second pass parses the rest of the directives (**Deny** or **Allow**). The third pass applies to all requests which do not match either of the first two.

Note that all **Allow** and **Deny** directives are processed, unlike a typical firewall, where only the first match is used. The last match is effective (also unlike a typical firewall). Additionally, the order in which lines appear in the configuration files is not significant -- all **Allow** lines are processed as one group, all **Deny** lines are considered as another, and the default state is considered by itself.

*Ordering* is one of:

### **Allow, Deny**

First, all **Allow** directives are evaluated; at least one must match, or the request is rejected. Next, all **Deny** directives are evaluated. If any matches, the request is rejected. Last, any requests which do not match an **Allow** or a **Deny** directive are denied by default.

## Deny, Allow

First, all [Deny](#) directives are evaluated; if any match, the request is denied **unless** it also matches an [Allow](#) directive. Any requests which do not match any [Allow](#) or [Deny](#) directives are permitted.

## Mutual-failure

This order has the same effect as Order Allow, Deny and is deprecated in its favor.

Keywords may only be separated by a comma; *no whitespace* is allowed between them.

Match	Allow,Deny result	Deny,Allow result
Match Allow only	Request allowed	Request allowed
Match Deny only	Request denied	Request denied
No match	Default to second directive: Denied	Default to second directive: Allowed
Match both Allow & Deny	Final match controls: Denied	Final match controls: Allowed

In the following example, all hosts in the example.org domain are allowed access; all other hosts are denied access.

```
Order Deny,Allow
Deny from all
Allow from example.org
```

In the next example, all hosts in the example.org domain are allowed access, except for the hosts which are in the foo.example.org subdomain, who are denied access. All hosts not in the example.org domain are denied access because the default

state is to [Deny](#) access to the server.

```
Order Allow,Deny
Allow from example.org
Deny from foo.example.org
```

On the other hand, if the [Order](#) in the last example is changed to [Deny,Allow](#), all hosts will be allowed access. This happens because, regardless of the actual ordering of the directives in the configuration file, the [Allow from example.org](#) will be evaluated last and will override the [Deny from foo.example.org](#). All hosts not in the [example.org](#) domain will also be allowed access because the default state is [Allow](#).

The presence of an [Order](#) directive can affect access to a part of the server even in the absence of accompanying [Allow](#) and [Deny](#) directives because of its effect on the default access state. For example,

```
<Directory "/www">
    Order Allow,Deny
</Directory>
```

will [Deny](#) all access to the [/www](#) directory because the default access state is set to [Deny](#).

The [Order](#) directive controls the order of access directive processing only within each phase of the server's configuration processing. This implies, for example, that an [Allow](#) or [Deny](#) directive occurring in a [<Location>](#) section will always be evaluated after an [Allow](#) or [Deny](#) directive occurring in a [<Directory>](#) section or [.htaccess](#) file, regardless of the setting of the [Order](#) directive. For details on the merging of

configuration sections, see the documentation on [How Directory, Location and Files sections work](#).

### **Merging of configuration sections**

When any directive provided by this module is used in a new configuration section, no directives provided by this module are inherited from previous configuration sections.



## Security Directives

<b>Description:</b>	Interaction between host-level access control and user authentication
<b>Syntax:</b>	Satisfy Any All
<b>Default:</b>	Satisfy All
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_access_compat
<b>Compatibility:</b>	Influenced by <a href="#">&lt;Limit&gt;</a> and <a href="#">&lt;LimitExcept&gt;</a> in version 2.0.51 and later

Access policy if both [Allow](#) and [Require](#) used. The parameter can be either All or Any. This directive is only useful if access to a particular area is being restricted by both username/password *and* client host address. In this case the default behavior (All) is to require that the client passes the address access restriction *and* enters a valid username and password. With the Any option the client will be granted access if they either pass the host restriction or enter a valid username and password. This can be used to password restrict an area, but to let clients from particular addresses in without prompting for a password.

For example, if you wanted to let people on your network have unrestricted access to a portion of your website, but require that people outside of your network provide a password, you could use a configuration similar to the following:

```
Require valid-user
Allow from 192.168.1
Satisfy Any
```

Another frequent use of the [Satisfy](#) directive is to relax access

restrictions for a subdirectory:

```
<Directory "/var/www/private">
    Require valid-user
</Directory>

<Directory "/var/www/private/public">
    Allow from all
    Satisfy Any
</Directory>
```

In the above example, authentication will be required for the `/var/www/private` directory, but will not be required for the `/var/www/private/public` directory.

Since version 2.0.51 [Satisfy](#) directives can be restricted to particular methods by [<Limit>](#) and [<LimitExcept>](#) sections.

### Merging of configuration sections

When any directive provided by this module is used in a new configuration section, no directives provided by this module are inherited from previous configuration sections.

### See also

- [Allow](#)
- [Require](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_actions

```
┆  
┆ CGI .  
┆ Base  
┆ actions_module  
┆ mod_actions.c
```

```
┆ Action MIME content type CGI  
┆ Script CGI .  
┆
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[mod\\_cgi](#)

[CGI](#)



```

: content-type CGI
: Action action-type cgi-script
  [virtual]
: , , directory, .htaccess
Override : FileInfo
: Base
: mod_actions
: virtual 2.1

```

```

action-type cgi-script . cgi-scrip
ScriptAlias AddHandler CGI URL.
type MIME content type . PATH_INFO
PATH_TRANSLATED CGI URL .
REDIRECT_HANDLER .

```

```

# MIME content type :
Action image/gif /cgi-bin/images.cgi

#
AddHandler my-file-type .xyz
Action my-file-type /cgi-bin/program.cgi

```

```

MIME content type image/gif cgi
bin/images.cgi .

.xyz cgi /cgi-bin/program.c

```

In the second example, requests for files with a file extension of .xyz are handled instead by the specified cgi script /cgi-bin/program.cgi.

```

virtual . ,

```

```
<Location /news>  
  SetHandler news-handler  
  Action news-handler /cgi-bin/news.cgi virtual  
</Location>
```

- [AddHandler](#)



## Script

```
Script CGI .  
Script method cgi-script  
Script , , directory  
Script Base  
Script mod_actions
```

```
ScriptAlias AddHandler CGI URL.  
Script PATH_INFO PATH_TRANSLATED CGI URL
```

```
Script .  
Script put .
```

```
Script . CGI ,  
Script GET Script ( , foo.html?hi)
```

```
# <ISINDEX>  
Script GET /cgi-bin/search  
  
# CGI PUT  
Script PUT /~bob/put.cgi
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_alias

```
┆  
┆ , URL  
┆ Base  
┆ alias_module  
┆ mod_alias.c
```

```
URL .  
ScriptAlias URL . DocumentRo  
., ScriptAlias CGI .  
Redirect URL .  
.  
mod_alias URL .  
.
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

mod\_rewrite

URL



Alias Redirect  
( , <VirtualHost> ) Alias Redirect  
.

Redirect Alias . Redirect  
RedirectMatch Alias . Alias Redirect

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

/foo/bar Alias /foo Alias



```

: URL
: Alias URL-path file-path|directory-path
: ,
: Base
: mod_alias

```

Alias DocumentRoot  
 (% ) URL *directory-path* .

```

:
Alias /image /ftp/pub/image

```

http://myserver/image/foo.gif /ftp/pub/image/foo.gif

*url-path /* , URL / . , A  
 /usr/local/apache/icons/ url /icons .

<Directory> . <Directory>  
 , .( <Location>  
 URL .)

DocumentRoot Alias , .

```

:
Alias /image /ftp/pub/image
<Directory /ftp/pub/image>
  Order allow,deny
  Allow from all
</Directory>

```



## AliasMatch

```
: URL
: AliasMatch regex file-path|directory-path
: ,
: Base
: mod_alias
```

```
Alias , URL . U
, .
:
```

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



## Redirect

```
URL
Redirect [status] URL-path URL
, , directory, .htaccess
Override : FileInfo
Base
mod_alias
```

```
Redirect URL URL . URL ,
.(%) URL-path (%)
URL .
```

```
:
Redirect /service http://foo2.bar.com/service
```

```
http://myserver/service/foo.txt
http://foo2.bar.com/service/foo.txt .
```

```
Redirect Alias ScriptAlias . ,
.htaccess <Directory> URL-path
URL .
```

```
status , " (temporary)" (HTTP 302) . ,
.status HTTP :
```

```
permanent (301) .
```

```
temp (302) . .
```

```
seeother " (See Other)" (303) .
```

```
gone  
    " (Gone)" (410) .  
    .
```

```
status . 300 399  
, . , (http_protocol.c  
send_error_response ).
```

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



## RedirectMatch

<a href="#">:</a>	URL
<a href="#">:</a>	RedirectMatch [ <i>status</i> ] <i>regex</i> URL
<a href="#">:</a>	, , directory, .htaccess
<a href="#">Override :</a>	FileInfo
<a href="#">:</a>	Base
<a href="#">:</a>	mod_alias

[Redirect](#) , URL .  
URL , . ,  
JPEG :

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```



Redirect

```
URL  
RedirectPermanent URL-path URL  
, , directory, .htaccess  
Override : FileInfo  
Base  
mod_alias
```

( 301) . Redirect



## RedirectTemp

```
URL
RedirectTemp URL -path URL
, , directory, .htaccess
Override : FileInfo
Base
mod_alias
```

( 302) . Redirect



## ScriptAlias

```
: URL CGI
: ScriptAlias URL-path file-path|directory-path
: ,
: Base
: mod_alias
```

`ScriptAlias` Alias , mod  
script CGI . *URL-path* (%) URL  
.

```
:  
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

`http://myserver/cgi-bin/foo` `/web/cgi-`  
`bin/foo` .



## ScriptAliasMatch

```
ScriptAliasMatch URL CGI
ScriptAliasMatch regex file-path|directory-path
ScriptAliasMatch Base
ScriptAliasMatch mod_alias
```

```
ScriptAliasMatch URL CGI
ScriptAliasMatch Base
ScriptAliasMatch mod_alias
```

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



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_allowmethods

<b>Description:</b>	Easily restrict what HTTP methods can be used on the server
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	allowmethods_module
<b>Source File:</b>	mod_allowmethods.c

## Summary

This module makes it easy to restrict what HTTP methods can be used on a server. The most common configuration would be:

```
<Location "/">  
    AllowMethods GET POST OPTIONS  
</Location>
```



<b>Description:</b>	Restrict access to the listed HTTP methods
<b>Syntax:</b>	<code>AllowMethods reset [HTTP-method [HTTP-method] ...</code>
<b>Default:</b>	<code>AllowMethods reset</code>
<b>Context:</b>	directory
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_allowmethods</code>

The HTTP-methods are case sensitive and are generally, as per RFC, given in upper case. The GET and HEAD methods are treated as equivalent. The reset keyword can be used to turn off [mod\\_allowmethods](#) in a deeper nested context:

```
<Location "/svn">
  AllowMethods reset
</Location>
```

### Caution

The TRACE method cannot be denied by this module; use [TraceEnable](#) instead.

[mod\\_allowmethods](#) was written to replace the rather kludgy implementation of [Limit](#) and [LimitExcept](#).



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_asis

[: HTTP](#)

[: Base](#)

[: asis\\_module](#)

[: mod\\_asis.c](#)

HTTP

cgi\_nph

HTTP

mime\_type

httpd/send-as-is .



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[mod\\_headers](#)

[mod\\_cern\\_meta](#)





```
send-as-is .
```

```
AddHandler send-as-is asis
```

```
.asis . HTTP  
Status: . HTTP .
```

```
Status: 301 Now where did I leave that URL  
Location: http://xyz.abc.com/foo/bar.html  
Content-type: text/html
```

```
<html>  
<head>  
<title>Lame excuses'R'us</title>  
</head>  
<body>  
<h1>Fred's exceptionally wonderful page has moved to  
<a href="http://xyz.abc.com/foo/bar.html">Joe's</a> site.  
</h1>  
</body>  
</html>
```

```
:  
Date: Server: , .  
Last-Modified . .
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_auth\_basic

- Basic authentication
- Base
- auth\_basic\_module
- mod\_auth\_basic.c
- 2.1

(provider) HTTP Basic Authentication `mod_auth_digest`.  
HTTP Digest Authentication



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[AuthName](#)

[AuthType](#)



```

:
: AuthBasicAuthoritative On|Off
: AuthBasicAuthoritative On
: directory, .htaccess
Override : AuthConfig
: Base
: mod_auth_basic

```

```

AuthBasicAuthoritative Off
( modules.c ) .
, "Authentication Required ( )
.
Require
AuthBasicAuthoritative .
, "Authentication Require
. , NCSA .

```



## AuthBasicFake

<a href="#">:</a>	Fake basic authentication using the given expressions for username and password
<a href="#">:</a>	AuthBasicFake off username [password]
<a href="#">:</a>	none
<a href="#">:</a>	directory, .htaccess
<b>Override :</b>	AuthConfig
<a href="#">:</a>	Base
<a href="#">:</a>	mod_auth_basic
<a href="#">:</a>	Apache HTTP Server 2.4.5 and later

The documentation for this directive has not been translated yet. Please have a look at the English version.



```
AuthBasicProvider On|Off|provider-name
[provider-name] ...
AuthBasicProvider On
directory, .htaccess
Override : AuthConfig
Base
mod_auth_basic
```

AuthBasicProvider .
. mod\_authn\_file file

```
<Location /secure>
  AuthBasicProvider dbm
  AuthDBMType SDBM
  AuthDBMUserFile /www/etc/dbmpasswd
  Require valid-user
</Location>
```

mod\_authn\_dbm mod\_authn\_file .

Off .



<code>:</code>	Check passwords against the authentication providers as if Digest Authentication was in force instead of Basic Authentication.
<code>:</code>	<code>AuthBasicUseDigestAlgorithm MD5 Off</code>
<code>:</code>	<code>AuthBasicUseDigestAlgorithm Off</code>
<code>:</code>	directory, .htaccess
<b><u>Override :</u></b>	<code>AuthConfig</code>
<code>:</code>	<code>Base</code>
<code>:</code>	<code>mod_auth_basic</code>
<code>:</code>	Apache HTTP Server 2.4.7 and later

The documentation for this directive has not been translated yet. Please have a look at the English version.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_auth\_digest

[: MD5 Digest Authentication](#)

[: Experimental](#)

[: auth\\_digest\\_module](#)

[: mod\\_auth\\_digest.c](#)

HTTP Digest Authentication .



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[AuthName](#)

[AuthType](#)

[Require](#)

[Satisfy](#)



## Digest Authentication

MD5 Digest authentication . AuthType  
Basic [AuthBasicProvider](#) AuthType Digest  
[AuthDigestProvider](#) .  
URI [AuthDigestDomain](#) .

[htdigest](#) () .

```
:  
<Location /private/>  
  AuthType Digest  
  AuthName "private area"  
  AuthDigestDomain /private/ http://mirror.my.dom/private2/  
  
  AuthDigestProvider file  
  AuthUserFile /web/auth/.digest_pw  
  Require valid-user  
</Location>
```

Digest authentication Basic authentication , .  
2002 11 digest authentication [Amaya](#),  
[Konqueror](#), (Windows - "  
[Explorer](#) ") Mac OS X Windows [MS Internet](#)  
[Explorer](#), [Mozilla](#), [Netscape](#) 7, [Opera](#), [Safari](#) . [lynx](#)  
digest authentication . digest authentication basic  
authentication



Windows Internet Explorer Digest authentication  
GET RFC .

GET POST .

, 2.0.51 AuthDigestEnableQueryStringHack  
. AuthDigestEnableQueryStringHack M  
URI digest . .

### MSIE Digest Authentication :

BrowserMatch "MSIE" AuthDigestEnableQueryStringHack=On

BrowserMatch .



## AuthDigestAlgorithm

<a href="#">:</a>	digest authentication challenge response hash
<a href="#">:</a>	AuthDigestAlgorithm MD5 MD5-sess
<a href="#">:</a>	AuthDigestAlgorithm MD5
<a href="#">:</a>	directory, .htaccess
<b>Override :</b>	AuthConfig
<a href="#">:</a>	Experimental
<a href="#">:</a>	mod_auth_digest

**AuthDigestAlgorithm** challenge response hash

.

MD5-sess .



## AuthDigestDomain

<a href="#">:</a>	digest authentication URI
<a href="#">:</a>	AuthDigestDomain <i>URI</i> [ <i>URI</i> ] ...
<a href="#">:</a>	directory, .htaccess
<a href="#">Override :</a>	AuthConfig
<a href="#">:</a>	Experimental
<a href="#">:</a>	mod_auth_digest

```
AuthDigestDomain ( /
. URI . URI "" / .URI
(scheme), , ) URL URI.
, URI()
Authorization . , AuthDigestNcC
.
URI , ( ) /
```



AuthDigestNonceLifetime

```
#: nonce
#: AuthDigestNonceLifetime seconds
#: AuthDigestNonceLifetime 300
#: directory, .htaccess
Override : AuthConfig
#: Experimental
#: mod_auth_digest
```

```
AuthDigestNonceLifetime nonce .
nonce stale=true 401 . S
nonce . 10 . secc
nonce .
```



## AuthDigestProvider

```
AuthDigestProvider On|Off|provider-name [provider-name] ...
AuthDigestProvider On
directory, .htaccess
Override : AuthConfig
Experimental
mod_auth_digest
```

```
AuthDigestProvider .
. mod_authn_file file
mod_authn_dbm mod_authn_file .
Off .
```



## AuthDigestQop

<a href="#">:</a>	digest authentication (quality-of-protection)
.	
<a href="#">:</a>	AuthDigestQop none auth auth-int [auth auth-int]
<a href="#">:</a>	AuthDigestQop auth
<a href="#">:</a>	directory, .htaccess
<b>Override :</b>	AuthConfig
<a href="#">:</a>	Experimental
<a href="#">:</a>	mod_auth_digest

```
AuthDigestQop (quality-of-protection) . auth (/)
, auth-int (MD5 ) .
) RFC-2069 digest . auth auth-int
. challenge
.
```

```
auth-int .
```



## AuthDigestShmemSize

```
AuthDigestShmemSize size
AuthDigestShmemSize 1000
Experimental
mod_auth_digest
```

### AuthDigestShmemSize

AuthDigestShmemSize 0

size , K M KBytes MBytes

```
AuthDigestShmemSize 1048576
AuthDigestShmemSize 1024K
AuthDigestShmemSize 1M
```



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_auth_form`

<b>Description:</b>	Form authentication
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>auth_form_module</code>
<b>Source File:</b>	<code>mod_auth_form.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

### Warning

Form authentication depends on the [mod\\_session](#) modules, and these modules make use of HTTP cookies, and as such can fall victim to Cross Site Scripting attacks, or expose potentially private information to clients. Please ensure that the relevant risks have been taken into account before enabling the session functionality on your server.

This module allows the use of an HTML login form to restrict access by looking up users in the given providers. HTML forms require significantly more configuration than the alternatives, however an HTML login form can provide a much friendlier experience for end users.

HTTP basic authentication is provided by [mod\\_auth\\_basic](#), and HTTP digest authentication is provided by [mod\\_auth\\_digest](#). This module should be combined with at least one authentication module such as [mod\\_authn\\_file](#) and one authorization module such as [mod\\_authz\\_user](#).

Once the user has been successfully authenticated, the user's login details will be stored in a session provided by [mod\\_session](#).



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_session](#)

[AuthName](#)

[AuthType](#)

[Require](#)

[Authentication howto](#)



To protect a particular URL with [mod\\_auth\\_form](#), you need to decide where you will store your *session*, and you will need to decide what method you will use to authenticate. In this simple example, the login details will be stored in a session based on [mod\\_session\\_cookie](#), and authentication will be attempted against a file using [mod\\_authn\\_file](#). If authentication is unsuccessful, the user will be redirected to the form login page.

### Basic example

```
<Location "/admin">
  AuthFormProvider file
  AuthUserFile "conf/passwd"
  AuthType form
  AuthName "/admin"
  AuthFormLoginRequiredLocation "http://example.com/login.htm:

  Session On
  SessionCookieName session path=/

  Require valid-user
</Location>
```

The directive [AuthType](#) will enable the [mod\\_auth\\_form](#) authentication when set to the value *form*. The directives [AuthFormProvider](#) and [AuthUserFile](#) specify that usernames and passwords should be checked against the chosen file.

The directives [Session](#) and [SessionCookieName](#) session stored within an HTTP cookie on the browser. For more information on the different options for configuring a session, read the documentation for [mod\\_session](#).

You can optionally add a [SessionCryptoPassphrase](#) to create an encrypted session cookie. This required the additional module

mod\_session\_crypto be loaded.

In the simple example above, a URL has been protected by mod\_auth\_form, but the user has yet to be given an opportunity to enter their username and password. Options for doing so include providing a dedicated standalone login page for this purpose, or for providing the login page inline.



## Example Login

The login form can be hosted as a standalone page, or can be provided inline on the same page.

When configuring the login as a standalone page, unsuccessful authentication attempts should be redirected to a login form created by the website for this purpose, using the [AuthFormLoginRequiredLocation](#) directive. Typically this login page will contain an HTML form, asking the user to provide their username and password.

### Example login form

```
<form method="POST" action="/dologin.html">
  Username: <input type="text" name="httpd_username" value="" />
  Password: <input type="password" name="httpd_password" value="" />
  <input type="submit" name="login" value="Login" />
</form>
```

The part that does the actual login is handled by the *form-login-handler*. The action of the form should point at this handler, which is configured within Apache httpd as follows:

### Form login handler example

```
<Location "/dologin.html">
  SetHandler form-login-handler
  AuthFormLoginRequiredLocation "http://example.com/login.html"
  AuthFormLoginSuccessLocation "http://example.com/admin/index.html"
  AuthFormProvider file
  AuthUserFile "conf/passwd"
  AuthType form
  AuthName /admin
  Session On
  SessionCookieName session path=/
</Location>
```

The URLs specified by the [AuthFormLoginRequiredLocation](#) directive will typically point to a page explaining to the user that

their login attempt was unsuccessful, and they should try again. The [AuthFormLoginSuccessLocation](#) directive specifies the URL the user should be redirected to upon successful login.

Alternatively, the URL to redirect the user to on success can be embedded within the login form, as in the example below. As a result, the same *form-login-handler* can be reused for different areas of a website.

### Example login form with location

```
<form method="POST" action="/dologin.html">
  Username: <input type="text" name="httpd_username" value="" />
  Password: <input type="password" name="httpd_password" value="" />
  <input type="submit" name="login" value="Login" />
  <input type="hidden" name="httpd_location" value="http://exam
</form>
```



## Warning

A risk exists that under certain circumstances, the login form configured using inline login may be submitted more than once, revealing login credentials to the application running underneath. The administrator must ensure that the underlying application is properly secured to prevent abuse. If in doubt, use the standalone login configuration.

As an alternative to having a dedicated login page for a website, it is possible to configure `mod_auth_form` to authenticate users inline, without being redirected to another page. This allows the state of the current page to be preserved during the login attempt. This can be useful in a situation where a time limited session is in force, and the session times out in the middle of the user request. The user can be re-authenticated in place, and they can continue where they left off.

If a non-authenticated user attempts to access a page protected by `mod_auth_form` that isn't configured with a `AuthFormLoginRequiredLocation` directive, a `HTTP_UNAUTHORIZED` status code is returned to the browser indicating to the user that they are not authorized to view the page.

To configure inline authentication, the administrator overrides the error document returned by the `HTTP_UNAUTHORIZED` status code with a custom error document containing the login form, as follows:

### Basic inline example

```
AuthFormProvider file
ErrorDocument 401 "/login.shtml"
AuthUserFile "conf/passwd"
AuthType form
```

```
AuthName realm
AuthFormLoginRequiredLocation "http://example.com/login.html"
Session On
SessionCookieName session path=/"
```

The error document page should contain a login form with an empty action property, as per the example below. This has the effect of submitting the form to the original protected URL, without the page having to know what that URL is.

### Example inline login form

```
<form method="POST" action="">
  Username: <input type="text" name="httpd_username" value="" />
  Password: <input type="password" name="httpd_password" value="" />
  <input type="submit" name="login" value="Login" />
</form>
```

When the end user has filled in their login details, the form will make an HTTP POST request to the original password protected URL. [mod\\_auth\\_form](#) will intercept this POST request, and if HTML fields are found present for the username and password, the user will be logged in, and the original password protected URL will be returned to the user as a GET request.



A limitation of the inline login technique described above is that should an HTML form POST have resulted in the request to authenticate or reauthenticate, the contents of the original form posted by the browser will be lost. Depending on the function of the website, this could present significant inconvenience for the end user.

[mod\\_auth\\_form](#) addresses this by allowing the method and body of the original request to be embedded in the login form. If authentication is successful, the original method and body will be retried by Apache httpd, preserving the state of the original request.

To enable body preservation, add three additional fields to the login form as per the example below.

#### Example with body preservation

```
<form method="POST" action="">
  Username: <input type="text" name="httpd_username" value="" />
  Password: <input type="password" name="httpd_password" value="" />
  <input type="submit" name="login" value="Login" />

  <input type="hidden" name="httpd_method" value="POST" />
  <input type="hidden" name="httpd_mimetype" value="application/x-www-form-urlencoded" />
  <input type="hidden" name="httpd_body" value="name1=value1&name2=value2" />
</form>
```

How the method, mimetype and body of the original request are embedded within the login form will depend on the platform and technology being used within the website.

One option is to use the [mod\\_include](#) module along with the [KeptBodySize](#) directive, along with a suitable CGI script to embed the variables in the form.

Another option is to render the login form using a CGI script or other dynamic technology.

### CGI example

```
AuthFormProvider file
ErrorDocument 401 "/cgi-bin/login.cgi"
...
```



## Logging Out

To enable a user to log out of a particular session, configure a page to be handled by the *form-logout-handler*. Any attempt to access this URL will cause the username and password to be removed from the current session, effectively logging the user out.

By setting the [AuthFormLogoutLocation](#) directive, a URL can be specified that the browser will be redirected to on successful logout. This URL might explain to the user that they have been logged out, and give the user the option to log in again.

### Basic logout example

```
SetHandler form-logout-handler
AuthName realm
AuthFormLogoutLocation "http://example.com/loggedout.html"
Session On
SessionCookieName session path=/'
```

Note that logging a user out does not delete the session; it merely removes the username and password from the session. If this results in an empty session, the net effect will be the removal of that session, but this is not guaranteed. If you want to guarantee the removal of a session, set the [SessionMaxAge](#) directive to a small value, like 1 (setting the directive to zero would mean no session age limit).

### Basic session expiry example

```
SetHandler form-logout-handler
AuthFormLogoutLocation "http://example.com/loggedout.html"
Session On
SessionMaxAge 1
SessionCookieName session path=/'
```



Note that form submission involves URLEncoding the form data: in this case the username and password. You should therefore pick usernames and passwords that avoid characters that are URLEncoded in form submission, or you may get unexpected results.



<b>Description:</b>	Sets whether authorization and authentication are passed to lower level modules
<b>Syntax:</b>	AuthFormAuthoritative On Off
<b>Default:</b>	AuthFormAuthoritative On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form

Normally, each authorization module listed in [AuthFormProvider](#) will attempt to verify the user, and if the user is not found in any provider, access will be denied. Setting the [AuthFormAuthoritative](#) directive explicitly to Off allows for both authentication and authorization to be passed on to other non-provider-based modules if there is **no userID** or **rule** matching the supplied userID. This should only be necessary when combining [mod\\_auth\\_form](#) with third-party modules that are not configured with the [AuthFormProvider](#) directive. When using such modules, the order of processing is determined in the modules' source code and is not configurable.



<b>Description:</b>	The name of a form field carrying the body of the request to attempt on successful login
<b>Syntax:</b>	AuthFormBody <i>fieldname</i>
<b>Default:</b>	httpd_body
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormMethod](#) directive specifies the name of an HTML field which, if present, will contain the method of the request to submit should login be successful.

By populating the form with fields described by [AuthFormMethod](#), [AuthFormMimetype](#) and [AuthFormBody](#), a website can retry a request that may have been interrupted by the login screen, or by a session timeout.



<b>Description:</b>	Disable the CacheControl no-store header on the login page
<b>Syntax:</b>	AuthFormDisableNoStore <i>On Off</i>
<b>Default:</b>	AuthFormDisableNoStore Off
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The `AuthFormDisableNoStore` flag disables the sending of a `Cache-Control no-store` header with the error 401 page returned when the user is not yet logged in. The purpose of the header is to make it difficult for an `ecmascript` application to attempt to resubmit the login form, and reveal the username and password to the backend application. Disable at your own risk.



<b>Description:</b>	Fake a Basic Authentication header
<b>Syntax:</b>	<code>AuthFormFakeBasicAuth On Off</code>
<b>Default:</b>	<code>AuthFormFakeBasicAuth Off</code>
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_auth_form</code>
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The `AuthFormFakeBasicAuth` flag determines whether a Basic Authentication header will be added to the request headers. This can be used to expose the username and password to an underlying application, without the underlying application having to be aware of how the login was achieved.



<b>Description:</b>	The name of a form field carrying a URL to redirect to on successful login
<b>Syntax:</b>	AuthFormLocation <i>fieldname</i>
<b>Default:</b>	httpd_location
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormLocation](#) directive specifies the name of an HTML field which, if present, will contain a URL to redirect the browser to should login be successful.



<b>Description:</b>	The URL of the page to be redirected to should login be required
<b>Syntax:</b>	<code>AuthFormLoginRequiredLocation <i>url</i></code>
<b>Default:</b>	none
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later. The use of the expression parser has been added in 2.4.4.

The [AuthFormLoginRequiredLocation](#) directive specifies the URL to redirect to should the user not be authorised to view a page. The value is parsed using the [ap\\_expr](#) parser before being sent to the client. By default, if a user is not authorised to view a page, the HTTP response code HTTP\_UNAUTHORIZED will be returned with the page specified by the [ErrorDocument](#) directive. This directive overrides this default.

Use this directive if you have a dedicated login page to redirect users to.



<b>Description:</b>	The URL of the page to be redirected to should login be successful
<b>Syntax:</b>	<code>AuthFormLoginSuccessLocation url</code>
<b>Default:</b>	none
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later. The use of the expression parser has been added in 2.4.4.

The [AuthFormLoginSuccessLocation](#) directive specifies the URL to redirect to should the user have logged in successfully. The value is parsed using the [ap\\_expr](#) parser before being sent to the client. This directive can be overridden if a form field has been defined containing another URL using the [AuthFormLocation](#) directive.

Use this directive if you have a dedicated login URL, and you have not embedded the destination page in the login form.



<b>Description:</b>	The URL to redirect to after a user has logged out
<b>Syntax:</b>	<code>AuthFormLogoutLocation <i>uri</i></code>
<b>Default:</b>	none
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later. The use of the expression parser has been added in 2.4.4.

The `AuthFormLogoutLocation` directive specifies the URL of a page on the server to redirect to should the user attempt to log out. The value is parsed using the `ap_expr` parser before being sent to the client.

When a URI is accessed that is served by the handler `form-logout-handler`, the page specified by this directive will be shown to the end user. For example:

### Example

```
<Location "/logout">
  SetHandler form-logout-handler
  AuthFormLogoutLocation "http://example.com/loggedout.html"
  Session on
  #...
</Location>
```

An attempt to access the URI `/logout/` will result in the user being logged out, and the page `/loggedout.html` will be displayed. Make sure that the page `loggedout.html` is not password protected, otherwise the page will not be displayed.



<b>Description:</b>	The name of a form field carrying the method of the request to attempt on successful login
<b>Syntax:</b>	AuthFormMethod <i>fieldname</i>
<b>Default:</b>	httpd_method
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormMethod](#) directive specifies the name of an HTML field which, if present, will contain the method of the request to submit should login be successful.

By populating the form with fields described by [AuthFormMethod](#), [AuthFormMimetype](#) and [AuthFormBody](#), a website can retry a request that may have been interrupted by the login screen, or by a session timeout.



<b>Description:</b>	The name of a form field carrying the mimetype of the body of the request to attempt on successful login
<b>Syntax:</b>	AuthFormMimetype <i>fieldname</i>
<b>Default:</b>	httpd_mimetype
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormMethod](#) directive specifies the name of an HTML field which, if present, will contain the mimetype of the request to submit should login be successful.

By populating the form with fields described by [AuthFormMethod](#), [AuthFormMimetype](#) and [AuthFormBody](#), a website can retry a request that may have been interrupted by the login screen, or by a session timeout.



<b>Description:</b>	The name of a form field carrying the login password
<b>Syntax:</b>	AuthFormPassword <i>fieldname</i>
<b>Default:</b>	htpasswd_password
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormPassword](#) directive specifies the name of an HTML field which, if present, will contain the password to be used to log in.



<b>Description:</b>	Sets the authentication provider(s) for this location
<b>Syntax:</b>	AuthFormProvider <i>provider-name</i> [ <i>provider-name</i> ] ...
<b>Default:</b>	AuthFormProvider file
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form

The `AuthFormProvider` directive sets which provider is used to authenticate the users for this location. The default `file` provider is implemented by the [mod\\_authn\\_file](#) module. Make sure that the chosen provider module is present in the server.

### Example

```
<Location "/secure">
  AuthType form
  AuthName "private area"
  AuthFormProvider dbm
  AuthDBMType SDBM
  AuthDBMUserFile "/www/etc/dbmpasswd"
  Require valid-user
  #...
</Location>
```

Providers are implemented by [mod\\_authn\\_dbm](#), [mod\\_authn\\_file](#), [mod\\_authn\\_dbd](#), [mod\\_authnz\\_ldap](#) and [mod\\_authn\\_socache](#).



<b>Description:</b>	Bypass authentication checks for high traffic sites
<b>Syntax:</b>	<code>AuthFormSitePassphrase <i>secret</i></code>
<b>Default:</b>	none
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The `AuthFormSitePassphrase` directive specifies a passphrase which, if present in the user session, causes Apache httpd to bypass authentication checks for the given URL. It can be used on high traffic websites to reduce the load induced on authentication infrastructure.

The passphrase can be inserted into a user session by adding this directive to the configuration for the `form-login-handler`. The `form-login-handler` itself will always run the authentication checks, regardless of whether a passphrase is specified or not.

### Warning

If the session is exposed to the user through the use of `mod_session_cookie`, and the session is not protected with `mod_session_crypto`, the passphrase is open to potential exposure through a dictionary attack. Regardless of how the session is configured, ensure that this directive is not used within URL spaces where private user data could be exposed, or sensitive transactions can be conducted. Use at own risk.



<b>Description:</b>	The largest size of the form in bytes that will be parsed for the login details
<b>Syntax:</b>	<code>AuthFormSize size</code>
<b>Default:</b>	8192
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormSize](#) directive specifies the maximum size of the body of the request that will be parsed to find the login form.

If a login request arrives that exceeds this size, the whole request will be aborted with the HTTP response code `HTTP_REQUEST_TOO_LARGE`.

If you have populated the form with fields described by [AuthFormMethod](#), [AuthFormMimetype](#) and [AuthFormBody](#), you probably want to set this field to a similar size as the [KeptBodySize](#) directive.



<b>Description:</b>	The name of a form field carrying the login username
<b>Syntax:</b>	AuthFormUsername <i>fieldname</i>
<b>Default:</b>	httpd_username
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_form
<b>Compatibility:</b>	Available in Apache HTTP Server 2.3.0 and later

The [AuthFormUsername](#) directive specifies the name of an HTML field which, if present, will contain the username to be used to log in.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_authn\_anon

- :(anonymous)"
- :(Extension
- :(authn\_anon\_module
- :(mod\_authn\_anon.c
- :(2.1

```
    mod_auth_basic ( " 'anonymous'
) -ftp . .
() "
    URL / URL .
mod_auth_basic AuthBasicProvider anon
```



```
"" htpasswd- '(guest)'  
:
```

- .( [Anonymous\\_NoUserID](#))
- .( [Anonymous\\_MustGiveEmail](#))
- . '@' ' ' .  
([Anonymous\\_VerifyEmail](#))
- anonymous guest www test welcome ,  
. ([Anonymous](#))
- . ( [Anonymous](#)

```
<Directory /foo>  
  AuthName " 'anonymous' "  
  AuthType Basic  
  AuthBasicProvider file anon  
  AuthUserFile /path/to/your/.htpasswd  
  
  Anonymous_NoUserID off  
  Anonymous_MustGiveEmail on  
  Anonymous_VerifyEmail on  
  Anonymous_LogEmail on  
  Anonymous anonymous guest www test welcome  
  
  Order Deny,Allow  
  Allow from all  
  
  Require valid-user  
</Directory>
```





## Anonymous\_LogEmail

```
:  
: Anonymous_LogEmail On|Off  
: Anonymous_LogEmail On  
: directory, .htaccess  
Override : AuthConfig  
: Extension  
: mod_authn_anon
```

On ( ) " .





```
Anonymous_NoUserID On|Off
Anonymous_NoUserID Off
directory, .htaccess
Override : AuthConfig
Extension
mod_authn_anon
```

On ( ) .  
OK MS-Explorer .



```
:  
: Anonymous_VerifyEmail On|Off  
: Anonymous_VerifyEmail Off  
: directory, .htaccess  
Override : AuthConfig  
: Extension  
: mod_authn_anon
```

On " '@' "

[Anonymous\\_LogEmail](#) ).



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_auth_core`

<b>Description:</b>	Core Authentication
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>auth_core_module</code>
<b>Source File:</b>	<code>mod_auth_core.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

### Summary

This module provides core authentication capabilities to allow or deny access to portions of the web site. `mod_auth_core` provides directives that are common to all authentication providers.



Extended authentication providers can be created within the configuration file and assigned an alias name. The alias providers can then be referenced through the directives [AuthBasicProvider](#) or [AuthDigestProvider](#) in the same way as a base authentication provider. Besides the ability to create and alias an extended provider, it also allows the same extended authentication provider to be reference by multiple locations.

## Examples

This example checks for passwords in two different text files.

### Checking multiple text password files

```
# Check here first
<AuthnProviderAlias file file1>
  AuthUserFile "/www/conf/passwords1"
</AuthnProviderAlias>

# Then check here
<AuthnProviderAlias file file2>
  AuthUserFile "/www/conf/passwords2"
</AuthnProviderAlias>

<Directory "/var/web/pages/secure">
  AuthBasicProvider file1 file2

  AuthType Basic
  AuthName "Protected Area"
  Require valid-user
</Directory>
```

The example below creates two different ldap authentication provider aliases based on the ldap provider. This allows a single authenticated location to be serviced by multiple ldap hosts:

### Checking multiple LDAP servers

```
<AuthnProviderAlias ldap ldap-alias1>
  AuthLDAPBindDN cn=youruser,o=ctx
  AuthLDAPBindPassword yourpassword
```

```
    AuthLDAPURL ldap://ldap.host/o=ctx
</AuthnProviderAlias>
<AuthnProviderAlias ldap ldap-other-alias>
    AuthLDAPBindDN cn=yourotheruser,o=dev
    AuthLDAPBindPassword yourotherpassword
    AuthLDAPURL ldap://other.ldap.host/o=dev?cn
</AuthnProviderAlias>

Alias "/secure" "/webpages/secure"
<Directory "/webpages/secure">
    AuthBasicProvider ldap-other-alias ldap-alias1

    AuthType Basic
    AuthName "LDAP Protected Place"
    Require valid-user
    # Note that Require ldap-* would not work here, since the
    # AuthnProviderAlias does not provide the config to authori:
    # that are implemented in the same module as the authenticat
</Directory>
```



<b>Description:</b>	Authorization realm for use in HTTP authentication
<b>Syntax:</b>	AuthName <i>auth-domain</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_authn_core

This directive sets the name of the authorization realm for a directory. This realm is given to the client so that the user knows which username and password to send. [AuthName](#) takes a single argument; if the realm name contains spaces, it must be enclosed in quotation marks. It must be accompanied by [AuthType](#) and [Require](#) directives, and directives such as [AuthUserFile](#) and [AuthGroupFile](#) to work.

For example:

```
AuthName "Top Secret"
```

The string provided for the `AuthName` is what will appear in the password dialog provided by most browsers.

## See also

- [Authentication, Authorization, and Access Control](#)
- [mod\\_authz\\_core](#)



**Description:** Enclose a group of directives that represent an extension of a base authentication provider and referenced by the specified alias

**Syntax:** `<AuthnProviderAlias baseProvider Alias> ... </AuthnProviderAlias>`

**Context:** server config

**Status:** Base

**Module:** mod\_auth\_core

`<AuthnProviderAlias>` and `</AuthnProviderAlias>` are used to enclose a group of authentication directives that can be referenced by the alias name using one of the directives [AuthBasicProvider](#) or [AuthDigestProvider](#).

This directive has no affect on authorization, even for modules that provide both authentication and authorization.



<b>Description:</b>	Type of user authentication
<b>Syntax:</b>	AuthType None Basic Digest Form
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_core

This directive selects the type of user authentication for a directory. The authentication types available are None, Basic (implemented by [mod\\_auth\\_basic](#)), Digest (implemented by [mod\\_auth\\_digest](#)), and Form (implemented by [mod\\_auth\\_form](#)).

To implement authentication, you must also use the [AuthName](#) and [Require](#) directives. In addition, the server must have an authentication-provider module such as [mod\\_authn\\_file](#) and an authorization module such as [mod\\_authz\\_user](#).

The authentication type None disables authentication. When authentication is enabled, it is normally inherited by each subsequent [configuration section](#), unless a different authentication type is specified. If no authentication is desired for a subsection of an authenticated section, the authentication type None may be used; in the following example, clients may access the /www/docs/public directory without authenticating:

```
<Directory "/www/docs">
  AuthType Basic
  AuthName Documents
  AuthBasicProvider file
  AuthUserFile "/usr/local/apache/passwd/p
  Require valid-user
```

```
</Directory>

<Directory "/www/docs/public">
    AuthType None
    Require all granted
</Directory>
```

When disabling authentication, note that clients which have already authenticated against another portion of the server's document tree will typically continue to send authentication HTTP headers or cookies with each request, regardless of whether the server actually requires authentication for every resource.

## See also

- [Authentication, Authorization, and Access Control](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_auth_dbd`

<b>Description:</b>	User authentication using an SQL database
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>authn_dbd_module</code>
<b>Source File:</b>	<code>mod_authn_dbd.c</code>
<b>Compatibility:</b>	Available in Apache 2.1 and later

## Summary

This module provides authentication front-ends such as [mod\\_auth\\_digest](#) and [mod\\_auth\\_basic](#) to authenticate users by looking up users in SQL tables. Similar functionality is provided by, for example, [mod\\_authn\\_file](#).

This module relies on [mod\\_dbd](#) to specify the backend database driver and connection parameters, and manage the database connections.

When using [mod\\_auth\\_basic](#) or [mod\\_auth\\_digest](#), this module is invoked via the [AuthBasicProvider](#) or [AuthDigestProvider](#) with the `dbd` value.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[AuthName](#)

[AuthType](#)

[AuthBasicProvider](#)

[AuthDigestProvider](#)

[DBDriver](#)

[DBDParams](#)

[Password Formats](#)



Some users of DBD authentication in HTTPD 2.2/2.4 have reported that it imposes a problematic load on the database. This is most likely where an HTML page contains hundreds of objects (e.g. images, scripts, etc) each of which requires authentication. Users affected (or concerned) by this kind of problem should use [mod\\_auth\\_socache](#) to cache credentials and take most of the load off the database.



## Configuration Example

This simple example shows use of this module in the context of the Authentication and DBD frameworks.

```
# mod_dbd configuration
# UPDATED to include authentication caching
DBDriver pgsql
DBDParams "dbname=apacheauth user=apache pas

DBDMin 4
DBDKeep 8
DBDMax 20
DBDExptime 300

<Directory "/usr/www/myhost/private">
  # mod_authn_core and mod_auth_basic config
  # for mod_authn_dbd
  AuthType Basic
  AuthName "My Server"

  # To cache credentials, put socache ahead
  AuthBasicProvider socache dbd

  # Also required for caching: tell the cache
  AuthnCacheProvideFor dbd
  AuthnCacheContext my-server

  # mod_authz_core configuration
  Require valid-user

  # mod_authn_dbd SQL query to authenticate
  AuthDBDUserPWQuery "SELECT password FROM a
</Directory>
```



## Exposing Login Information

If httpd was built against [APR](#) version 1.3.0 or higher, then whenever a query is made to the database server, all column values in the first row returned by the query are placed in the environment, using environment variables with the prefix "AUTHENTICATE\_".

If a database query for example returned the username, full name and telephone number of a user, a CGI program will have access to this information without the need to make a second independent database query to gather this additional information.

This has the potential to dramatically simplify the coding and configuration required in some web applications.



<b>Description:</b>	SQL query to look up a password for a user
<b>Syntax:</b>	<code>AuthDBDUserPWQuery query</code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_authn_dbd</code>

The `AuthDBDUserPWQuery` specifies an SQL query to look up a password for a specified user. The user's ID will be passed as a single string parameter when the SQL query is executed. It may be referenced within the query statement using a `%s` format specifier.

```
AuthDBDUserPWQuery "SELECT password FROM aut
```

The first column value of the first row returned by the query statement should be a string containing the encrypted password. Subsequent rows will be ignored. If no rows are returned, the user will not be authenticated through `mod_authn_dbd`.

If `httpd` was built against `APR` version 1.3.0 or higher, any additional column values in the first row returned by the query statement will be stored as environment variables with names of the form `AUTHENTICATE_COLUMN`.

The encrypted password format depends on which authentication frontend (e.g. `mod_auth_basic` or `mod_auth_digest`) is being used. See [Password Formats](#) for more information.



<b>Description:</b>	SQL query to look up a password hash for a user and realm.
<b>Syntax:</b>	<code>AuthDBDUserRealmQuery <i>query</i></code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_authn_dbd</code>

The `AuthDBDUserRealmQuery` specifies an SQL query to look up a password for a specified user and realm in a digest authentication process. The user's ID and the realm, in that order, will be passed as string parameters when the SQL query is executed. They may be referenced within the query statement using %s format specifiers.

```
AuthDBDUserRealmQuery "SELECT password FROM
```

The first column value of the first row returned by the query statement should be a string containing the encrypted password. Subsequent rows will be ignored. If no rows are returned, the user will not be authenticated through `mod_authn_dbd`.

If httpd was built against `APR` version 1.3.0 or higher, any additional column values in the first row returned by the query statement will be stored as environment variables with names of the form `AUTHENTICATE_COLUMN`.

The encrypted password format depends on which authentication frontend (e.g. `mod_auth_basic` or `mod_auth_digest`) is being used. See [Password Formats](#) for more information.

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_authn\_dbm

- DBM
- Extension
- authn\_dbm\_module
- mod\_authn\_dbm.c
- 2.1

`mod_auth_digest mod_auth_basic dbm`  
`mod_authn_file`

`mod_auth_basic mod_auth_digest AuthBasicProvider`  
`AuthDigestProvider dbm`



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[AuthName](#)

[AuthType](#)

[AuthBasicProvider](#)

## AuthDigestProvider



## AuthDBMType

```
:  
: AuthDBMType default | SDBM | GDBM | NDBM | DB  
: AuthDBMType default  
: directory, .htaccess  
Override : AuthConfig  
: Extension  
: mod_authn_dbm
```



```
AuthDBMUserFile file-path
directory, .htaccess
Override : AuthConfig
Extension
mod_authn_dbm
```

AuthDBMUserFile DBM
path .

```
AuthDBMUserFile
AuthDBMUserFile
```

dbmopen NULL DBM
Netscape NULL

dbmmanage Perl . DBM



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_authn\_file

```
┆  
┆ Base  
┆ authn_file_module  
┆ mod_authn_file.c  
┆ 2.1
```

```
    mod_auth_digest mod_auth_basic  
    mod_authn_dbm .
```

```
mod_auth_basic mod_auth_digest AuthBasicProvider  
AuthDigestProvider file .
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[AuthBasicProvider](#)

[AuthDigestProvider](#)

[htpasswd](#)

[htdigest](#)



```
AuthUserFile file-path
directory, .htaccess
Override : AuthConfig
Base
mod_authn_file
```

AuthUserFile

```
ServerRoot
mod_authn_file
src/support htpasswd HTTP Basic
Authentication manpage :
username Filename . :
```

```
htpasswd -c Filename username
```

```
Filename username2 :
```

```
htpasswd Filename username2
```

AuthDBMUserFile

```
HTTP Digest Authentication htpasswd htdigest
. Digest Authentication Basic Authentication
```

```
AuthUserFile
```

, `AuthUserFile` .

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_auth\_socache

<b>Description:</b>	Manages a cache of authentication credentials to relieve the load on backends
<b>Status:</b>	Base
<b>Module Identifier:</b>	authn_socache_module
<b>Source File:</b>	mod_auth_socache.c
<b>Compatibility:</b>	Version 2.3 and later

### Summary

Maintains a cache of authentication credentials, so that a new backend lookup is not required for every authenticated request.



Some users of more heavyweight authentication such as SQL database lookups (`mod_authn_dbd`) have reported it putting an unacceptable load on their authentication provider. A typical case in point is where an HTML page contains hundreds of objects (images, scripts, stylesheets, media, etc), and a request to the page generates hundreds of effectively-immediate requests for authenticated additional contents.

`mod_auth_socache` provides a solution to this problem by maintaining a cache of authentication credentials.



The authentication cache should be used where authentication lookups impose a significant load on the server, or a backend or network. Authentication by file ([mod\\_authn\\_file](#)) or dbm ([mod\\_authn\\_dbm](#)) are unlikely to benefit, as these are fast and lightweight in their own right (though in some cases, such as a network-mounted file, caching may be worthwhile). Other providers such as SQL or LDAP based authentication are more likely to benefit, particularly where there is an observed performance issue. Amongst the standard modules, [mod\\_authnz\\_ldap](#) manages its own cache, so only [mod\\_authn\\_dbd](#) will usually benefit from this cache.

The basic rules to cache for a provider are:

1. Include the provider you're caching for in an [AuthnCacheProvideFor](#) directive.
2. List *socache* ahead of the provider you're caching for in your [AuthBasicProvider](#) or [AuthDigestProvider](#) directive.

A simple usage example to accelerate [mod\\_authn\\_dbd](#) using dbm as a cache engine:

```
#AuthnCacheSOCache is optional.  If specific
AuthnCacheSOCache dbm
<Directory "/usr/www/myhost/private">
  AuthType Basic
  AuthName "Cached Authentication Example"
  AuthBasicProvider socache dbd
  AuthDBDUserPWQuery "SELECT password FROM
  AuthnCacheProvideFor dbd
  Require valid-user
  #Optional
  AuthnCacheContext dbd-authn-example
</Directory>
```



## Caching Authn Modules

Module developers should note that their modules must be enabled for caching with `mod_auth_socache`. A single optional API function `ap_auth_cache_store` is provided to cache credentials a provider has just looked up or generated. Usage examples are available in [r957072](#), in which three authn providers are enabled for caching.



<b>Description:</b>	Specify a context string for use in the cache key
<b>Syntax:</b>	AuthnCacheContext <i>directory server custom-string</i>
<b>Default:</b>	directory
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_socache

This directive specifies a string to be used along with the supplied username (and realm in the case of Digest Authentication) in constructing a cache key. This serves to disambiguate identical usernames serving different authentication areas on the server.

Two special values for this are *directory*, which uses the directory context of the request as a string, and *server* which uses the virtual host name.

The default is *directory*, which is also the most conservative setting. This is likely to be less than optimal, as it (for example) causes *\$app-base*, *\$app-base/images*, *\$app-base/scripts* and *\$app-base/media* each to have its own separate cache key. A better policy is to name the **AuthnCacheContext** for the password provider: for example a *htpasswd* file or database table.

Contexts can be shared across different areas of a server, where credentials are shared. However, this has potential to become a vector for cross-site or cross-application security breaches, so this directive is not permitted in *.htaccess* contexts.



<b>Description:</b>	Enable Authn caching configured anywhere
<b>Syntax:</b>	AuthnCacheEnable
<b>Context:</b>	server config
<b>Override:</b>	None
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_socache

This directive is not normally necessary: it is implied if authentication caching is enabled anywhere in *httpd.conf*. However, if it is not enabled anywhere in *httpd.conf* it will by default not be initialised, and is therefore not available in a *.htaccess* context. This directive ensures it is initialised so it can be used in *.htaccess*.



<b>Description:</b>	Specify which authn provider(s) to cache for
<b>Syntax:</b>	AuthnCacheProvideFor <i>authn-provider</i> [...]
<b>Default:</b>	None
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_auth_socache

This directive specifies an authentication provider or providers to cache for. Credentials found by a provider not listed in an AuthnCacheProvideFor directive will not be cached.

For example, to cache credentials found by `mod_authn_dbd` or by a custom provider *myprovider*, but leave those looked up by lightweight providers like file or dbm lookup alone:

```
AuthnCacheProvideFor dbd myprovider
```



<b>Description:</b>	Select socache backend provider to use
<b>Syntax:</b>	<code>AuthnCacheS0Cache <i>provider-name[:provider-args]</i></code>
<b>Context:</b>	server config
<b>Override:</b>	None
<b>Status:</b>	Base
<b>Module:</b>	mod_authn_socache
<b>Compatibility:</b>	Optional provider arguments are available in Apache HTTP Server 2.4.7 and later

This is a server-wide setting to select a provider for the [shared object cache](#), followed by optional arguments for that provider. Some possible values for *provider-name* are "dbm", "dc", "memcache", or "shmcb", each subject to the appropriate module being loaded. If not set, your platform's default will be used.



<b>Description:</b>	Set a timeout for cache entries
<b>Syntax:</b>	<code>AuthnCacheTimeout</code> <i>timeout</i> (seconds)
<b>Default:</b>	300 (5 minutes)
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_authn_socache

Cacheing authentication data can be a security issue, though short-term cacheing is unlikely to be a problem. Typically a good solution is to cache credentials for as long as it takes to relieve the load on a backend, but no longer, though if changes to your users and passwords are infrequent then a longer timeout may suit you. The default 300 seconds (5 minutes) is both cautious and ample to keep the load on a backend such as dbd (SQL database queries) down.

This should not be confused with session timeout, which is an entirely separate issue. However, you may wish to check your session-management software for whether cached credentials can "accidentally" extend a session, and bear it in mind when setting your timeout.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_authnz\_fcgi

<b>Description:</b>	Allows a FastCGI authorizer application to handle Apache httpd authentication and authorization
<b>Status:</b>	Extension
<b>Module Identifier:</b>	authnz_fcgi_module
<b>Source File:</b>	mod_authnz_fcgi.c
<b>Compatibility:</b>	Available in version 2.4.10 and later

## Summary

This module allows FastCGI authorizer applications to authenticate users and authorize access to resources. It supports generic FastCGI authorizers which participate in a single phase for authentication and authorization as well as Apache httpd-specific authenticators and authorizers which participate in one or both phases.

FastCGI authorizers can authenticate using user id and password, such as for Basic authentication, or can authenticate using arbitrary mechanisms.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

## [Authentication, Authorization, and Access Control](#)

[mod\\_auth\\_basic](#)

[fcgistarter](#)

[mod\\_proxy\\_fcgi](#)



The invocation modes for FastCGI authorizers supported by this module are distinguished by two characteristics, *type* and *auth mechanism*.

*Type* is simply `authn` for authentication, `authz` for authorization, or `authnz` for combined authentication and authorization.

*Auth mechanism* refers to the Apache httpd configuration mechanisms and processing phases, and can be `AuthBasicProvider`, `Require`, or `check_user_id`. The first two of these correspond to the directives used to enable participation in the appropriate processing phase.

Descriptions of each mode:

### ***Type* `authn`, *mechanism* `AuthBasicProvider`**

In this mode, `FCGI_ROLE` is set to `AUTHORIZER` and `FCGI_APACHE_ROLE` is set to `AUTHENTICATOR`. The application must be defined as provider type *authn* using [AuthnzFcgiDefineProvider](#) and enabled with [AuthBasicProvider](#). When invoked, the application is expected to authenticate the client using the provided user id and password. Example application:

```
#!/usr/bin/perl
use FCGI;
my $request = FCGI::Request();
while ($request->Accept() >= 0) {
    die if $ENV{'FCGI_APACHE_ROLE'} ne "
    die if $ENV{'FCGI_ROLE'}          ne "
    die if !$ENV{'REMOTE_PASSWD'};
    die if !$ENV{'REMOTE_USER'};
```

```

print STDERR "This text is written t

if ( ($ENV{'REMOTE_USER'} } eq "foo"
      $ENV{'REMOTE_PASSWD'} eq "bar" )
    print "Status: 200\n";
    print "Variable-AUTHN_1: authn_0
    print "Variable-AUTHN_2: authn_0
    print "\n";
}
else {
    print "Status: 401\n\n";
}
}

```

Example configuration:

```

AuthnzFcgiDefineProvider authn FooAuthn
<Location "/protected/">
  AuthType Basic
  AuthName "Restricted"
  AuthBasicProvider FooAuthn
  Require ...
</Location>

```

### **Type authz, *mechanism* Require**

In this mode, FCGI\_ROLE is set to AUTHORIZER and FCGI\_APACHE\_ROLE is set to AUTHORIZER. The application must be defined as provider type *authz* using [AuthnzFcgiDefineProvider](#). When invoked, the application is expected to authorize the client using the provided user id and other request data. Example application:

```

#!/usr/bin/perl
use FCGI;
my $request = FCGI::Request();
while ($request->Accept() >= 0) {
    die if $ENV{'FCGI_APACHE_ROLE'} ne "
    die if $ENV{'FCGI_ROLE'}          ne "
    die if $ENV{'REMOTE_PASSWD'};

    print STDERR "This text is written t

    if ($ENV{'REMOTE_USER'} eq "foo1") {
        print "Status: 200\n";
        print "Variable-AUTHZ_1: authz_0
        print "Variable-AUTHZ_2: authz_0
        print "\n";
    }
    else {
        print "Status: 403\n\n";
    }
}

```

Example configuration:

```

AuthnzFcgiDefineProvider authz FooAuthz
<Location "/protected/">
    AuthType ...
    AuthName ...
    AuthBasicProvider ...
    Require FooAuthz
</Location>

```

**Type authnz, mechanism AuthBasicProvider + Require**

In this mode, which supports the web server-agnostic FastCGI AUTHORIZER protocol, FCGI\_ROLE is set to AUTHORIZER and FCGI\_APACHE\_ROLE is not set. The application must be defined as provider type *authnz* using [AuthnzFcgiDefineProvider](#). The application is expected to handle both authentication and authorization in the same invocation using the user id, password, and other request data. The invocation occurs during the Apache httpd API authentication phase. If the application returns 200 and the same provider is invoked during the authorization phase (via [Require](#)), mod\_authnz\_fcgi will return success for the authorization phase without invoking the application. Example application:

```
#!/usr/bin/perl
use FCGI;
my $request = FCGI::Request();
while ($request->Accept() >= 0) {
    die if $ENV{'FCGI_APACHE_ROLE'};
    die if $ENV{'FCGI_ROLE'} ne "AUTHORI
    die if !$ENV{'REMOTE_PASSWD'};
    die if !$ENV{'REMOTE_USER'};

    print STDERR "This text is written t

    if ( ($ENV{'REMOTE_USER'} } eq "foo"
        $ENV{'REMOTE_PASSWD'} } eq "bar" &
        $ENV{'REQUEST_URI'} } =~ m%/bar/.*)
        print "Status: 200\n";
        print "Variable-AUTHNZ_1: authnz
        print "Variable-AUTHNZ_2: authnz
        print "\n";
    }
    else {
```

```
        print "Status: 401\n\n";
    }
}
```

Example configuration:

```
AuthnzFcgiDefineProvider authnz FooAuthn
<Location "/protected/">
    AuthType Basic
    AuthName "Restricted"
    AuthBasicProvider FooAuthnz
    Require FooAuthnz
</Location>
```

### **Type *authn*, *mechanism* *check\_user\_id***

In this mode, `FCGI_ROLE` is set to `AUTHORIZER` and `FCGI_APACHE_ROLE` is set to `AUTHENTICATOR`. The application must be defined as provider type *authn* using [AuthnzFcgiDefineProvider](#).

[AuthnzFcgiCheckAuthnProvider](#) specifies when it is called. Example application:

```
#!/usr/bin/perl
use FCGI;
my $request = FCGI::Request();
while ($request->Accept() >= 0) {
    die if $ENV{'FCGI_APACHE_ROLE'} ne "
    die if $ENV{'FCGI_ROLE'} ne "AUTHORI

    # This authorizer assumes that the R
    # AuthnzFcgiCheckAuthnProvider is On
    die if !$ENV{'REMOTE_PASSWD'};
```

```

die if !$ENV{'REMOTE_USER'};

print STDERR "This text is written t

if ( ($ENV{'REMOTE_USER' } eq "foo"
      $ENV{'REMOTE_PASSWD'} eq "bar" )
    print "Status: 200\n";
    print "Variable-AUTHNZ_1: authnz
    print "Variable-AUTHNZ_2: authnz
    print "\n";
}
else {
    print "Status: 401\n\n";
    # If a response body is written
    # the client.
}
}

```

Example configuration:

```

AuthnzFcgiDefineProvider authn FooAuthn
<Location "/protected/">
    AuthType ...
    AuthName ...
    AuthnzFcgiCheckAuthnProvider FooAuthn
                                Authorita
                                RequireBa
                                UserExpr

    Require ...
</Location>

```



## Additional Examples

1. If your application supports the separate authentication and authorization roles (AUTHENTICATOR and AUTHORIZER), define separate providers as follows, even if they map to the same application:

```
AuthnzFcgiDefineProvider authn FooAuthn
AuthnzFcgiDefineProvider authz FooAuthz
```

Specify the authn provider on [AuthBasicProvider](#) and the authz provider on [Require](#):

```
AuthType Basic
AuthName "Restricted"
AuthBasicProvider FooAuthn
Require FooAuthz
```

2. If your application supports the generic AUTHORIZER role (authentication and authorizer in one invocation), define a single provider as follows:

```
AuthnzFcgiDefineProvider authnz FooAuthn
```

Specify the authnz provider on both [AuthBasicProvider](#) and [Require](#):

```
AuthType Basic
AuthName "Restricted"
AuthBasicProvider FooAuthnz
Require FooAuthnz
```



---

The following are potential features which are not currently implemented:

### **Apache httpd access checker**

The Apache httpd API *access check* phase is a separate phase from authentication and authorization. Some other FastCGI implementations implement this phase, which is denoted by the setting of `FCGI_APACHE_ROLE` to `ACCESS_CHECKER`.

### **Local (Unix) sockets or pipes**

Only TCP sockets are currently supported.

### **Support for `mod_auth_socache`**

`mod_auth_socache` interaction should be implemented for applications which participate in Apache httpd-style authentication.

### **Support for digest authentication using `AuthDigestProvider`**

This is expected to be a permanent limitation as there is no authorizer flow for retrieving a hash.

### **Application process management**

This is expected to be permanently out of scope for this module. Application processes must be controlled by other means. For example, [fcgistarter](#) can be used to start them.

### **`AP_AUTH_INTERNAL_PER_URI`**

All providers are currently registered as `AP_AUTH_INTERNAL_PER_CONF`, which means that checks are not performed again for internal subrequests with the same access control configuration as the initial request.

### **Protocol data charset conversion**

If `mod_authnz_fcgi` runs in an EBCDIC compilation environment, all FastCGI protocol data is written in EBCDIC

and expected to be received in EBCDIC.

### **Multiple requests per connection**

Currently the connection to the FastCGI authorizer is closed after every phase of processing. For example, if the authorizer handles separate *authn* and *authz* phases then two connections will be used.

### **URI Mapping**

URIs from clients can't be mapped, such as with the **ProxyPass** used with FastCGI responders.



## Logging

1. Processing errors are logged at log level `error` and higher.
2. Messages written by the application are logged at log level `warn`.
3. General messages for debugging are logged at log level `debug`.
4. Environment variables passed to the application are logged at log level `trace2`. The value of the `REMOTE_PASSWD` variable will be obscured, but **any other sensitive data will be visible in the log**.
5. All I/O between the module and the FastCGI application, including all environment variables, will be logged in printable and hex format at log level `trace5`. **All sensitive data will be visible in the log**.

`LogLevel` can be used to configure a log level specific to `mod_authz_fcgi`. For example:

```
LogLevel info authz_fcgi:trace8
```



<b>Description:</b>	Enables a FastCGI application to handle the check_authn authentication hook.
<b>Syntax:</b>	AuthnzFcgiCheckAuthnProvider <i>provider-name</i>   None <i>option</i> ...
<b>Default:</b>	none
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_fcgi

This directive is used to enable a FastCGI authorizer to handle a specific processing phase of authentication or authorization.

Some capabilities of FastCGI authorizers require enablement using this directive instead of [AuthBasicProvider](#):

- Non-Basic authentication; generally, determining the user id of the client and returning it from the authorizer; see the UserExpr option below
- Selecting a custom response code; for a non-200 response from the authorizer, the code from the authorizer will be the status of the response
- Setting the body of a non-200 response; if the authorizer provides a response body with a non-200 response, that body will be returned to the client; up to 8192 bytes of text are supported

### ***provider-name***

This is the name of a provider defined with [AuthnzFcgiDefineProvider](#).

### **None**

Specify None to disable a provider enabled with this directive in an outer scope, such as in a parent directory.

## *option*

The following options are supported:

### **Authoritative On|Off (default On)**

This controls whether or not other modules are allowed to run when this module has a FastCGI authorizer configured and it fails the request.

### **DefaultUser *userid***

When the authorizer returns success and UserExpr is configured and evaluates to an empty string (e.g., authorizer didn't return a variable), this value will be used as the user id. This is typically used when the authorizer has a concept of guest, or unauthenticated, users and guest users are mapped to some specific user id for logging and other purposes.

### **RequireBasicAuth On|Off (default Off)**

This controls whether or not Basic auth is required before passing the request to the authorizer. If required, the authorizer won't be invoked without a user id and password; 401 will be returned for a request without that.

### **UserExpr *expr* (no default)**

When Basic authentication isn't provided by the client and the authorizer determines the user, this expression, evaluated after calling the authorizer, determines the user. The expression follows [ap\\_expr syntax](#) and must resolve to a string. A typical use is to reference a `Variable-XXX` setting returned by the authorizer using an option like `UserExpr "%{reqenv:XXX}"`. If this option is specified and the user id can't be retrieved using the expression after a successful authentication, the request will be rejected with a 500 error.



<b>Description:</b>	Defines a FastCGI application as a provider for authentication and/or authorization
<b>Syntax:</b>	<code>AuthnzFcgiDefineProvider <i>type</i> <i>provider-name</i> <i>backend-address</i></code>
<b>Default:</b>	none
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_authnz_fcgi</code>

This directive is used to define a FastCGI application as a provider for a particular phase of authentication or authorization.

### ***type***

This must be set to *authn* for authentication, *authz* for authorization, or *authnz* for a generic FastCGI authorizer which performs both checks.

### ***provider-name***

This is used to assign a name to the provider which is used in other directives such as [AuthBasicProvider](#) and [Require](#).

### ***backend-address***

This specifies the address of the application, in the form `fcgi://hostname:port/`. The application process(es) must be managed independently, such as with [fcgistarter](#).



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_authnz_ldap`

<b>Description:</b>	Allows an LDAP directory to be used to store the database for HTTP Basic authentication.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>authnz_ldap_module</code>
<b>Source File:</b>	<code>mod_authnz_ldap.c</code>
<b>Compatibility:</b>	Available in version 2.1 and later

## Summary

This module allows authentication front-ends such as `mod_auth_basic` to authenticate users through an ldap directory.

`mod_authnz_ldap` supports the following features:

- Known to support the [OpenLDAP SDK](#) (both 1.x and 2.x), [Novell LDAP SDK](#) and the [iPlanet \(Netscape\) SDK](#).
- Complex authorization policies can be implemented by representing the policy with LDAP filters.
- Uses extensive caching of LDAP operations via [mod\\_ldap](#).
- Support for LDAP over SSL (requires the Netscape SDK) or TLS (requires the OpenLDAP 2.x SDK or Novell LDAP SDK).

When using `mod_auth_basic`, this module is invoked via the [AuthBasicProvider](#) directive with the `ldap` value.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_ldap](#)

[mod\\_auth\\_basic](#)

[mod\\_authz\\_user](#)

[mod\\_authz\\_groupfile](#)



- 
- [General caveats](#)
  - [Operation](#)
    - [The Authentication Phase](#)
    - [The Authorization Phase](#)
  - [The Require Directives](#)
    - [Require ldap-user](#)
    - [Require ldap-group](#)
    - [Require ldap-dn](#)
    - [Require ldap-attribute](#)
    - [Require ldap-filter](#)
  - [Examples](#)
  - [Using TLS](#)
  - [Using SSL](#)
  - [Exposing Login Information](#)
  - [Using Active Directory](#)
  - [Using Microsoft FrontPage with mod\\_authnz\\_ldap](#)
    - [How It Works](#)
    - [Caveats](#)



This module caches authentication and authorization results based on the configuration of [mod\\_ldap](#). Changes made to the backing LDAP server will not be immediately reflected on the HTTP Server, including but not limited to user lockouts/revocations, password changes, or changes to group memberships. Consult the directives in [mod\\_ldap](#) for details of the cache tunables.



There are two phases in granting access to a user. The first phase is authentication, in which the [mod\\_authnz\\_ldap](#) authentication provider verifies that the user's credentials are valid. This is also called the *search/bind* phase. The second phase is authorization, in which [mod\\_authnz\\_ldap](#) determines if the authenticated user is allowed access to the resource in question. This is also known as the *compare* phase.

[mod\\_authnz\\_ldap](#) registers both an `authn_ldap` authentication provider and an `authz_ldap` authorization handler. The `authn_ldap` authentication provider can be enabled through the [AuthBasicProvider](#) directive using the `ldap` value. The `authz_ldap` handler extends the [Require](#) directive's authorization types by adding `ldap-user`, `ldap-dn` and `ldap-group` values.

## The Authentication Phase

During the authentication phase, [mod\\_authnz\\_ldap](#) searches for an entry in the directory that matches the username that the HTTP client passes. If a single unique match is found, then [mod\\_authnz\\_ldap](#) attempts to bind to the directory server using the DN of the entry plus the password provided by the HTTP client. Because it does a search, then a bind, it is often referred to as the *search/bind* phase. Here are the steps taken during the *search/bind* phase.

1. Generate a search filter by combining the attribute and filter provided in the [AuthLDAPURL](#) directive with the username passed by the HTTP client.
2. Search the directory using the generated filter. If the search does not return exactly one entry, deny or decline access.
3. Fetch the distinguished name of the entry retrieved from the search and attempt to bind to the LDAP server using that DN

and the password passed by the HTTP client. If the bind is unsuccessful, deny or decline access.

The following directives are used during the search/bind phase

<a href="#"><u>AuthLDAPURL</u></a>	Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to use.
<a href="#"><u>AuthLDAPBindDN</u></a>	An optional DN to bind with during the search phase.
<a href="#"><u>AuthLDAPBindPassword</u></a>	An optional password to bind with during the search phase.

## The Authorization Phase

During the authorization phase, [mod\\_authnz\\_ldap](#) attempts to determine if the user is authorized to access the resource. Many of these checks require [mod\\_authnz\\_ldap](#) to do a compare operation on the LDAP server. This is why this phase is often referred to as the compare phase. [mod\\_authnz\\_ldap](#) accepts the following [Require](#) directives to determine if the credentials are acceptable:

- Grant access if there is a [Require ldap-user](#) directive, and the username in the directive matches the username passed by the client.
- Grant access if there is a [Require ldap-dn](#) directive, and the DN in the directive matches the DN fetched from the LDAP directory.
- Grant access if there is a [Require ldap-group](#) directive, and the DN fetched from the LDAP directory (or the username passed by the client) occurs in the LDAP group or, potentially, in one of its sub-groups.

- Grant access if there is a [Require ldap-attribute](#) directive, and the attribute fetched from the LDAP directory matches the given value.
- Grant access if there is a [Require ldap-filter](#) directive, and the search filter successfully finds a single user object that matches the dn of the authenticated user.
- otherwise, deny or decline access

Other [Require](#) values may also be used which may require loading additional authorization modules.

- Grant access to all successfully authenticated users if there is a [Require valid-user](#) directive. (requires [mod\\_authz\\_user](#))
- Grant access if there is a [Require group](#) directive, and [mod\\_authz\\_groupfile](#) has been loaded with the [AuthGroupFile](#) directive set.
- others...

[mod\\_authnz\\_ldap](#) uses the following directives during the compare phase:

<a href="#">AuthLDAPURL</a>	The attribute specified in the URL is used in compare operations for the <a href="#">Require ldap-user</a> operation.
<a href="#">AuthLDAPCompareDNOnServer</a>	Determines the behavior of the <a href="#">Require ldap-dn</a> directive.
<a href="#">AuthLDAPGroupAttribute</a>	Determines the attribute to use for comparisons in the <a href="#">Require ldap-group</a> directive.
<a href="#">AuthLDAPGroupAttributeIsDN</a>	Specifies whether to use the

user DN or the username when doing comparisons for the Require ldap-group directive.

AuthLDAPMaxSubGroupDepth

Determines the maximum depth of sub-groups that will be evaluated during comparisons in the Require ldap-group directive.

AuthLDAPSubGroupAttribute

Determines the attribute to use when obtaining sub-group members of the current group during comparisons in the Require ldap-group directive.

AuthLDAPSubGroupClass

Specifies the LDAP objectClass values used to identify if queried directory objects really are group objects (as opposed to user objects) during the Require ldap-group directive's sub-group processing.



## THE REQUIRE DIRECTIVES

Apache's [Require](#) directives are used during the authorization phase to ensure that a user is allowed to access a resource. `mod_authnz_ldap` extends the authorization types with `ldap-user`, `ldap-dn`, `ldap-group`, `ldap-attribute` and `ldap-filter`. Other authorization types may also be used but may require that additional authorization modules be loaded.

Since v2.4.8, [expressions](#) are supported within the LDAP require directives.

### Require ldap-user

The `Require ldap-user` directive specifies what usernames can access the resource. Once `mod_authnz_ldap` has retrieved a unique DN from the directory, it does an LDAP compare operation using the username specified in the `Require ldap-user` to see if that username is part of the just-fetched LDAP entry. Multiple users can be granted access by putting multiple usernames on the line, separated with spaces. If a username has a space in it, then it must be surrounded with double quotes. Multiple users can also be granted access by using multiple `Require ldap-user` directives, with one user per line. For example, with a [AuthLDAPURL](#) of `ldap://ldap/o=Example?cn` (i.e., `cn` is used for searches), the following `Require` directives could be used to restrict access:

```
Require ldap-user "Barbara Jenson"  
Require ldap-user "Fred User"  
Require ldap-user "Joe Manager"
```

Because of the way that `mod_authnz_ldap` handles this directive, Barbara Jenson could sign on as *Barbara Jenson*, *Babs Jenson* or any other `cn` that she has in her LDAP entry. Only the

single `Require ldap-user` line is needed to support all values of the attribute in the user's entry.

If the `uid` attribute was used instead of the `cn` attribute in the URL above, the above three lines could be condensed to

```
Require ldap-user bjenson fuser jmanager
```

## Require ldap-group

This directive specifies an LDAP group whose members are allowed access. It takes the distinguished name of the LDAP group. Note: Do not surround the group name with quotes. For example, assume that the following entry existed in the LDAP directory:

```
dn: cn=Administrators, o=Example
objectClass: groupOfUniqueNames
uniqueMember: cn=Barbara Jenson, o=Example
uniqueMember: cn=Fred User, o=Example
```

The following directive would grant access to both Fred and Barbara:

```
Require ldap-group cn=Administrators, o=Exar
```

Members can also be found within sub-groups of a specified LDAP group if [AuthLDAPMaxSubGroupDepth](#) is set to a value greater than 0. For example, assume the following entries exist in the LDAP directory:

```
dn: cn=Employees, o=Example
objectClass: groupOfUniqueNames
uniqueMember: cn=Managers, o=Example
uniqueMember: cn=Administrators, o=Example
uniqueMember: cn=Users, o=Example
```

```
dn: cn=Managers, o=Example
objectClass: groupOfUniqueNames
uniqueMember: cn=Bob Ellis, o=Example
uniqueMember: cn=Tom Jackson, o=Example

dn: cn=Administrators, o=Example
objectClass: groupOfUniqueNames
uniqueMember: cn=Barbara Jenson, o=Example
uniqueMember: cn=Fred User, o=Example

dn: cn=Users, o=Example
objectClass: groupOfUniqueNames
uniqueMember: cn=Allan Jefferson, o=Example
uniqueMember: cn=Paul Tilley, o=Example
uniqueMember: cn=Temporary Employees, o=Example

dn: cn=Temporary Employees, o=Example
objectClass: groupOfUniqueNames
uniqueMember: cn=Jim Swenson, o=Example
uniqueMember: cn=Elliot Rhodes, o=Example
```

The following directives would allow access for Bob Ellis, Tom Jackson, Barbara Jenson, Fred User, Allan Jefferson, and Paul Tilley but would not allow access for Jim Swenson, or Elliot Rhodes (since they are at a sub-group depth of 2):

```
Require ldap-group cn=Employees, o=Example
AuthLDAPMaxSubGroupDepth 1
```

Behavior of this directive is modified by the [AuthLDAPGroupAttribute](#), [AuthLDAPGroupAttributeIsDN](#), [AuthLDAPMaxSubGroupDepth](#), [AuthLDAPSubGroupAttribute](#), and [AuthLDAPSubGroupClass](#) directives.

## Require ldap-dn

The `Require ldap-dn` directive allows the administrator to grant

access based on distinguished names. It specifies a DN that must match for access to be granted. If the distinguished name that was retrieved from the directory server matches the distinguished name in the `Require ldap-dn`, then authorization is granted.

Note: do not surround the distinguished name with quotes.

The following directive would grant access to a specific DN:

```
Require ldap-dn cn=Barbara Jenson, o=Example
```

Behavior of this directive is modified by the [AuthLDAPCompareDNOnServer](#) directive.

## Require ldap-attribute

The `Require ldap-attribute` directive allows the administrator to grant access based on attributes of the authenticated user in the LDAP directory. If the attribute in the directory matches the value given in the configuration, access is granted.

The following directive would grant access to anyone with the attribute `employeeType = active`

```
Require ldap-attribute "employeeType=active"
```

Multiple attribute/value pairs can be specified on the same line separated by spaces or they can be specified in multiple `Require ldap-attribute` directives. The effect of listing multiple attribute/values pairs is an OR operation. Access will be granted if any of the listed attribute values match the value of the corresponding attribute in the user object. If the value of the attribute contains a space, only the value must be within double

quotes.

The following directive would grant access to anyone with the city attribute equal to "San Jose" or status equal to "Active"

```
Require ldap-attribute city="San Jose" "stat
```

## Require ldap-filter

The `Require ldap-filter` directive allows the administrator to grant access based on a complex LDAP search filter. If the dn returned by the filter search matches the authenticated user dn, access is granted.

The following directive would grant access to anyone having a cell phone and is in the marketing department

```
Require ldap-filter "&(cell=*)(department=ma
```

The difference between the `Require ldap-filter` directive and the `Require ldap-attribute` directive is that `ldap-filter` performs a search operation on the LDAP directory using the specified search filter rather than a simple attribute comparison. If a simple attribute comparison is all that is required, the comparison operation performed by `ldap-attribute` will be faster than the search operation used by `ldap-filter` especially within a large directory.



- Grant access to anyone who exists in the LDAP directory, using their UID for searches.

```
AuthLDAPURL "ldap://ldap1.example.com:389  
Require valid-user
```

- The next example is the same as above; but with the fields that have useful defaults omitted. Also, note the use of a redundant LDAP server.

```
AuthLDAPURL "ldap://ldap1.example.com ldap://ldap2.example.com  
Require valid-user
```

- The next example is similar to the previous one, but it uses the common name instead of the UID. Note that this could be problematical if multiple people in the directory share the same cn, because a search on cn **must** return exactly one entry. That's why this approach is not recommended: it's a better idea to choose an attribute that is guaranteed unique in your directory, such as uid.

```
AuthLDAPURL "ldap://ldap.example.com/ou=users  
Require valid-user
```

- Grant access to anybody in the Administrators group. The users must authenticate using their UID.

```
AuthLDAPURL ldap://ldap.example.com/o=Example.com  
Require ldap-group cn=Administrators, o=Example.com
```

- Grant access to anybody in the group whose name matches the hostname of the virtual host. In this example an [expression](#) is used to build the filter.

```
AuthLDAPURL ldap://ldap.example.com/o=Ex
Require ldap-group cn=%{SERVER_NAME}, o=
```

- The next example assumes that everyone at Example who carries an alphanumeric pager will have an LDAP attribute of `qpagePagerID`. The example will grant access only to people (authenticated via their UID) who have alphanumeric pagers:

```
AuthLDAPURL ldap://ldap.example.com/o=Ex
Require valid-user
```

- The next example demonstrates the power of using filters to accomplish complicated administrative requirements. Without filters, it would have been necessary to create a new LDAP group and ensure that the group's members remain synchronized with the pager users. This becomes trivial with filters. The goal is to grant access to anyone who has a pager, plus grant access to Joe Manager, who doesn't have a pager, but does need to access the same resource:

```
AuthLDAPURL ldap://ldap.example.com/o=Ex
Require valid-user
```

This last may look confusing at first, so it helps to evaluate what the search filter will look like based on who connects, as shown below. If Fred User connects as `fuser`, the filter would

look like

```
(&( |(qpagePagerID=*)(uid=jmanager))(uid=fuser))
```

The above search will only succeed if *fuser* has a pager.  
When Joe Manager connects as *jmanager*, the filter looks like

```
(&( |(qpagePagerID=*)(uid=jmanager))(uid=jmanager))
```

The above search will succeed whether *jmanager* has a pager or not.



To use TLS, see the [mod\\_ldap](#) directives [LDAPTrustedClientCert](#), [LDAPTrustedGlobalCert](#) and [LDAPTrustedMode](#).

An optional second parameter can be added to the [AuthLDAPURL](#) to override the default connection type set by [LDAPTrustedMode](#). This will allow the connection established by an *ldap://* Url to be upgraded to a secure connection on the same port.



To use SSL, see the [mod\\_ldap](#) directives [LDAPTrustedClientCert](#), [LDAPTrustedGlobalCert](#) and [LDAPTrustedMode](#).

To specify a secure LDAP server, use *ldaps://* in the [AuthLDAPURL](#) directive, instead of *ldap://*.



## Exposing LDAP Information

when this module performs *authentication*, ldap attributes specified in the `authldapurl` directive are placed in environment variables with the prefix "AUTHENTICATE\_".

when this module performs *authorization*, ldap attributes specified in the `authldapurl` directive are placed in environment variables with the prefix "AUTHORIZE\_".

If the attribute field contains the username, common name and telephone number of a user, a CGI program will have access to this information without the need to make a second independent LDAP query to gather this additional information.

This has the potential to dramatically simplify the coding and configuration required in some web applications.



## Configuring Active Directory

An Active Directory installation may support multiple domains at the same time. To distinguish users between domains, an identifier called a User Principle Name (UPN) can be added to a user's entry in the directory. This UPN usually takes the form of the user's account name, followed by the domain components of the particular domain, for example *somebody@nz.example.com*.

You may wish to configure the `mod_authnz_ldap` module to authenticate users present in any of the domains making up the Active Directory forest. In this way both *somebody@nz.example.com* and *someone@au.example.com* can be authenticated using the same query at the same time.

To make this practical, Active Directory supports the concept of a Global Catalog. This Global Catalog is a read only copy of selected attributes of all the Active Directory servers within the Active Directory forest. Querying the Global Catalog allows all the domains to be queried in a single query, without the query spanning servers over potentially slow links.

If enabled, the Global Catalog is an independent directory server that runs on port 3268 (3269 for SSL). To search for a user, do a subtree search for the attribute *userPrincipalName*, with an empty search root, like so:

```
AuthLDAPBindDN apache@example.com
AuthLDAPBindPassword password
AuthLDAPURL ldap://10.0.0.1:3268/?userPrinc:
```

Users will need to enter their User Principal Name as a login, in the form *somebody@nz.example.com*.



Normally, FrontPage uses FrontPage-web-specific user/group files (i.e., the `mod_authn_file` and `mod_authz_groupfile` modules) to handle all authentication. Unfortunately, it is not possible to just change to LDAP authentication by adding the proper directives, because it will break the *Permissions* forms in the FrontPage client, which attempt to modify the standard text-based authorization files.

Once a FrontPage web has been created, adding LDAP authentication to it is a matter of adding the following directives to every `.htaccess` file that gets created in the web

```
AuthLDAPURL      "the url"  
AuthGroupFile    "mygroupfile"  
Require group    "mygroupfile"
```

## How It Works

FrontPage restricts access to a web by adding the `Require valid-user` directive to the `.htaccess` files. The `Require valid-user` directive will succeed for any user who is valid as far as *LDAP is concerned*. This means that anybody who has an entry in the LDAP directory is considered a valid user, whereas FrontPage considers only those people in the local user file to be valid. By substituting the `ldap-group` with group file authorization, Apache is allowed to consult the local user file (which is managed by FrontPage) - instead of LDAP - when handling authorizing the user.

Once directives have been added as specified above, FrontPage users will be able to perform all management operations from the FrontPage client.

## Caveats

- When choosing the LDAP URL, the attribute to use for authentication should be something that will also be valid for putting into a [mod\\_authn\\_file](#) user file. The user ID is ideal for this.
- When adding users via FrontPage, FrontPage administrators should choose usernames that already exist in the LDAP directory (for obvious reasons). Also, the password that the administrator enters into the form is ignored, since Apache will actually be authenticating against the password in the LDAP database, and not against the password in the local user file. This could cause confusion for web administrators.
- Apache must be compiled with [mod\\_auth\\_basic](#), [mod\\_authn\\_file](#) and [mod\\_authz\\_groupfile](#) in order to use FrontPage support. This is because Apache will still use the [mod\\_authz\\_groupfile](#) group file for determine the extent of a user's access to the FrontPage web.
- The directives must be put in the .htaccess files. Attempting to put them inside [<Location>](#) or [<Directory>](#) directives won't work. This is because [mod\\_authnz\\_ldap](#) has to be able to grab the [AuthGroupFile](#) directive that is found in FrontPage .htaccess files so that it knows where to look for the valid user list. If the [mod\\_authnz\\_ldap](#) directives aren't in the same .htaccess file as the FrontPage directives, then the hack won't work, because [mod\\_authnz\\_ldap](#) will never get a chance to process the .htaccess file, and won't be able to find the FrontPage-managed user file.



<b>Description:</b>	Specifies the prefix for environment variables set during authorization
<b>Syntax:</b>	AuthLDAPAuthorizePrefix <i>prefix</i>
<b>Default:</b>	AuthLDAPAuthorizePrefix AUTHORIZE_
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.6 and later

This directive allows you to override the prefix used for environment variables set during LDAP authorization. If *AUTHENTICATE\_* is specified, consumers of these environment variables see the same information whether LDAP has performed authentication, authorization, or both.

#### Note

No authorization variables are set when a user is authorized on the basis of `Require valid-user`.



<b>Description:</b>	Determines if other authentication providers are used when a user can be mapped to a DN but the server cannot successfully bind with the user's credentials.
<b>Syntax:</b>	AuthLDAPBindAuthoritative off on
<b>Default:</b>	AuthLDAPBindAuthoritative on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

By default, subsequent authentication providers are only queried if a user cannot be mapped to a DN, but not if the user can be mapped to a DN and their password cannot be verified with an LDAP bind. If `AuthLDAPBindAuthoritative` is set to `off`, other configured authentication modules will have a chance to validate the user if the LDAP bind (with the current user's credentials) fails for any reason.

This allows users present in both LDAP and `AuthUserFile` to authenticate when the LDAP server is available but the user's account is locked or password is otherwise unusable.

## See also

- [AuthUserFile](#)
- [AuthBasicProvider](#)



<b>Description:</b>	Optional DN to use in binding to the LDAP server
<b>Syntax:</b>	AuthLDAPBindDN <i>distinguished-name</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

An optional DN used to bind to the server when searching for entries. If not provided, [mod\\_authnz\\_ldap](#) will use an anonymous bind.



<b>Description:</b>	Password used in conjunction with the bind DN
<b>Syntax:</b>	AuthLDAPBindPassword <i>password</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	exec: was added in 2.4.5.

A bind password to use in conjunction with the bind DN. Note that the bind password is probably sensitive data, and should be properly protected. You should only use the [AuthLDAPBindDN](#) and [AuthLDAPBindPassword](#) if you absolutely need them to search the directory.

If the value begins with `exec:` the resulting command will be executed and the first line returned to standard output by the program will be used as the password.

```
#Password used as-is
AuthLDAPBindPassword secret

#Run /path/to/program to get my password
AuthLDAPBindPassword exec:/path/to/program

#Run /path/to/otherProgram and provide argur
AuthLDAPBindPassword "exec:/path/to/otherPro
```



<b>Description:</b>	Language to charset conversion configuration file
<b>Syntax:</b>	AuthLDAPCharsetConfig <i>file-path</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

The `AuthLDAPCharsetConfig` directive sets the location of the language to charset conversion configuration file. *File-path* is relative to the `ServerRoot`. This file specifies the list of language extensions to character sets. Most administrators use the provided `charset.conv` file, which associates common language extensions to character sets.

The file contains lines in the following format:

```
Language-Extension charset [Language-String] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (#) are ignored.



<b>Description:</b>	Use the authenticated user's credentials to perform authorization comparisons
<b>Syntax:</b>	AuthLDAPCompareAsUser on off
<b>Default:</b>	AuthLDAPCompareAsUser off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.6 and later

When set, and [mod\\_authnz\\_ldap](#) has authenticated the user, LDAP comparisons for authorization use the queried distinguished name (DN) and HTTP basic authentication password of the authenticated user instead of the servers configured credentials.

The *ldap-attribute*, *ldap-user*, and *ldap-group* (single-level only) authorization checks use comparisons.

This directive only has effect on the comparisons performed during nested group processing when [AuthLDAPSearchAsUser](#) is also enabled.

This directive should only be used when your LDAP server doesn't accept anonymous comparisons and you cannot use a dedicated [AuthLDAPBindDN](#).

## See also

- [AuthLDAPInitialBindAsUser](#)
- [AuthLDAPSearchAsUser](#)



<b>Description:</b>	Use the LDAP server to compare the DN's
<b>Syntax:</b>	AuthLDAPCompareDNOnServer on off
<b>Default:</b>	AuthLDAPCompareDNOnServer on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

When set, [mod\\_authnz\\_ldap](#) will use the LDAP server to compare the DN's. This is the only foolproof way to compare DN's. [mod\\_authnz\\_ldap](#) will search the directory for the DN specified with the [Require dn](#) directive, then, retrieve the DN and compare it with the DN retrieved from the user entry. If this directive is not set, [mod\\_authnz\\_ldap](#) simply does a string comparison. It is possible to get false negatives with this approach, but it is much faster. Note the [mod\\_ldap](#) cache can speed up DN comparison in most situations.



<b>Description:</b>	When will the module de-reference aliases
<b>Syntax:</b>	AuthLDAPDereferenceAliases never   searching   finding   always
<b>Default:</b>	AuthLDAPDereferenceAliases always
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

This directive specifies when `mod_authnz_ldap` will de-reference aliases during LDAP operations. The default is `always`.



## AuthLDAPGroupAttribute Directive

<b>Description:</b>	LDAP attributes used to identify the user members of groups.
<b>Syntax:</b>	AuthLDAPGroupAttribute <i>attribute</i>
<b>Default:</b>	AuthLDAPGroupAttribute member uniquemember
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

This directive specifies which LDAP attributes are used to check for user members within groups. Multiple attributes can be used by specifying this directive multiple times. If not specified, then [mod\\_authnz\\_ldap](#) uses the member and uniquemember attributes.



<b>Description:</b>	Use the DN of the client username when checking for group membership
<b>Syntax:</b>	AuthLDAPGroupAttributeIsDN on off
<b>Default:</b>	AuthLDAPGroupAttributeIsDN on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

When set on, this directive says to use the distinguished name of the client username when checking for group membership. Otherwise, the username will be used. For example, assume that the client sent the username bjenson, which corresponds to the LDAP DN cn=Babs Jenson, o=Example. If this directive is set, [mod\\_authnz\\_ldap](#) will check if the group has cn=Babs Jenson, o=Example as a member. If this directive is not set, then [mod\\_authnz\\_ldap](#) will check if the group has bjenson as a member.



<b>Description:</b>	Determines if the server does the initial DN lookup using the basic authentication users' own username, instead of anonymously or with hard-coded credentials for the server
<b>Syntax:</b>	AuthLDAPInitialBindAsUser off on
<b>Default:</b>	AuthLDAPInitialBindAsUser off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.6 and later

By default, the server either anonymously, or with a dedicated user and password, converts the basic authentication username into an LDAP distinguished name (DN). This directive forces the server to use the verbatim username and password provided by the incoming user to perform the initial DN search.

If the verbatim username can't directly bind, but needs some cosmetic transformation, see [AuthLDAPInitialBindPattern](#).

This directive should only be used when your LDAP server doesn't accept anonymous searches and you cannot use a dedicated [AuthLDAPBindDN](#).

### **Not available with authorization-only**

This directive can only be used if this module authenticates the user, and has no effect when this module is used exclusively for authorization.

## **See also**

- [AuthLDAPInitialBindPattern](#)
- [AuthLDAPBindDN](#)
- [AuthLDAPCompareAsUser](#)
- [AuthLDAPSearchAsUser](#)



<b>Description:</b>	Specifies the transformation of the basic authentication username to be used when binding to the LDAP server to perform a DN lookup
<b>Syntax:</b>	<code>AuthLDAPInitialBindPattern <i>regex substitution</i></code>
<b>Default:</b>	<code>AuthLDAPInitialBindPattern (.*) \$1</code> (remote username used verbatim)
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.6 and later

If [AuthLDAPInitialBindAsUser](#) is set to *ON*, the basic authentication username will be transformed according to the regular expression and substitution arguments.

The regular expression argument is compared against the current basic authentication username. The substitution argument may contain backreferences, but has no other variable interpolation.

This directive should only be used when your LDAP server doesn't accept anonymous searches and you cannot use a dedicated [AuthLDAPBindDN](#).

```
AuthLDAPInitialBindPattern (.+) $1@example.com
```

```
AuthLDAPInitialBindPattern (.+) cn=$1,dc=example.com
```

### **Not available with authorization-only**

This directive can only be used if this module authenticates the user, and has no effect when this module is used exclusively for authorization.

### **debugging**

The substituted DN is recorded in the environment variable *LDAP\_BINDASUSER*. If the regular expression does not match the input, the verbatim username is used.

### **See also**

- [AuthLDAPInitialBindAsUser](#)
- [AuthLDAPBindDN](#)



<b>Description:</b>	Specifies the maximum sub-group nesting depth that will be evaluated before the user search is discontinued.
<b>Syntax:</b>	AuthLDAPMaxSubGroupDepth <i>Number</i>
<b>Default:</b>	AuthLDAPMaxSubGroupDepth 10
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.0 and later

When this directive is set to a non-zero value X combined with use of the `Require ldap-group someGroupDN` directive, the provided user credentials will be searched for as a member of the someGroupDN directory object or of any group member of the current group up to the maximum nesting level X specified by this directive.

See the [Require ldap-group](#) section for a more detailed example.

### Nested groups performance

When `AuthLDAPSubGroupAttribute` overlaps with `AuthLDAPGroupAttribute` (as it does by default and as required by common LDAP schemas), uncached searching for subgroups in large groups can be very slow. If you use large, non-nested groups, set `AuthLDAPMaxSubGroupDepth` to zero.



**Description:** Use the value of the attribute returned during the user query to set the REMOTE\_USER environment variable

**Syntax:** AuthLDAPRemoteUserAttribute uid

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod\_authnz\_ldap

If this directive is set, the value of the REMOTE\_USER environment variable will be set to the value of the attribute specified. Make sure that this attribute is included in the list of attributes in the AuthLDAPUrl definition, otherwise this directive will have no effect. This directive, if present, takes precedence over [AuthLDAPRemoteUserIsDN](#). This directive is useful should you want people to log into a website using an email address, but a backend application expects the username as a userid.



<b>Description:</b>	Use the DN of the client username to set the REMOTE_USER environment variable
<b>Syntax:</b>	AuthLDAPRemoteUserIsDN on off
<b>Default:</b>	AuthLDAPRemoteUserIsDN off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

If this directive is set to on, the value of the REMOTE\_USER environment variable will be set to the full distinguished name of the authenticated user, rather than just the username that was passed by the client. It is turned off by default.



<b>Description:</b>	Use the authenticated user's credentials to perform authorization searches
<b>Syntax:</b>	AuthLDAPSearchAsUser on off
<b>Default:</b>	AuthLDAPSearchAsUser off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.6 and later

When set, and [mod\\_authnz\\_ldap](#) has authenticated the user, LDAP searches for authorization use the queried distinguished name (DN) and HTTP basic authentication password of the authenticated user instead of the servers configured credentials.

The *ldap-filter* and *ldap-dn* authorization checks use searches.

This directive only has effect on the comparisons performed during nested group processing when [AuthLDAPCompareAsUser](#) is also enabled.

This directive should only be used when your LDAP server doesn't accept anonymous searches and you cannot use a dedicated [AuthLDAPBindDN](#).

## See also

- [AuthLDAPInitialBindAsUser](#)
- [AuthLDAPCompareAsUser](#)



<b>Description:</b>	Specifies the attribute labels, one value per directive line, used to distinguish the members of the current group that are groups.
<b>Syntax:</b>	AuthLDAPSubGroupAttribute <i>attribute</i>
<b>Default:</b>	AuthLDAPSubgroupAttribute member uniquemember
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.0 and later

An LDAP group object may contain members that are users and members that are groups (called nested or sub groups). The [AuthLDAPSubGroupAttribute](#) directive identifies the labels of group members and the [AuthLDAPGroupAttribute](#) directive identifies the labels of the user members. Multiple attributes can be used by specifying this directive multiple times. If not specified, then [mod\\_authnz\\_ldap](#) uses the member and uniqueMember attributes.



<b>Description:</b>	Specifies which LDAP objectClass values identify directory objects that are groups during sub-group processing.
<b>Syntax:</b>	AuthLDAPSubGroupClass <i>LdapObjectClass</i>
<b>Default:</b>	AuthLDAPSubGroupClass groupOfNames groupOfUniqueNames
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in version 2.3.0 and later

An LDAP group object may contain members that are users and members that are groups (called nested or sub groups). The [AuthLDAPSubGroupAttribute](#) directive identifies the labels of members that may be sub-groups of the current group (as opposed to user members). The [AuthLDAPSubGroupClass](#) directive specifies the LDAP objectClass values used in verifying that these potential sub-groups are in fact group objects. Verified sub-groups can then be searched for more user or sub-group members. Multiple attributes can be used by specifying this directive multiple times. If not specified, then [mod\\_authnz\\_ldap](#) uses the groupOfNames and groupOfUniqueNames values.



<b>Description:</b>	URL specifying the LDAP search parameters
<b>Syntax:</b>	AuthLDAPUrl <i>url</i> [ <i>NONE</i>   <i>SSL</i>   <i>TLS</i>   <i>STARTTLS</i> ]
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

An RFC 2255 URL which specifies the LDAP search parameters to use. The syntax of the URL is

```
ldap://host:port/basedn?attribute?scope?filter
```

If you want to specify more than one LDAP URL that Apache should try in turn, the syntax is:

```
AuthLDAPUrl "ldap://ldap1.example.com ldap2
```

**Caveat:** *If you specify multiple servers, you need to enclose the entire URL string in quotes; otherwise you will get an error: "AuthLDAPURL takes one argument, URL to define LDAP connection.."* You can of course use search parameters on each of these.

## ldap

For regular ldap, use the string ldap. For secure LDAP, use ldaps instead. Secure LDAP is only available if Apache was linked to an LDAP library with SSL support.

## host:port

The name/port of the ldap server (defaults to localhost:389 for ldap, and localhost:636 for

ldaps). To specify multiple, redundant LDAP servers, just list all servers, separated by spaces. `mod_authnz_ldap` will try connecting to each server in turn, until it makes a successful connection. If multiple ldap servers are specified, then entire LDAP URL must be encapsulated in double quotes.

Once a connection has been made to a server, that connection remains active for the life of the `httpd` process, or until the LDAP server goes down.

If the LDAP server goes down and breaks an existing connection, `mod_authnz_ldap` will attempt to re-connect, starting with the primary server, and trying each redundant server in turn. Note that this is different than a true round-robin search.

### **basedn**

The DN of the branch of the directory where all searches should start from. At the very least, this must be the top of your directory tree, but could also specify a subtree in the directory.

### **attribute**

The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute will be used, no matter how many are provided. If no attributes are provided, the default is to use `uid`. It's a good idea to choose an attribute that will be unique across all entries in the subtree you will be using. All attributes listed will be put into the environment with an `AUTHENTICATE_` prefix for use by other modules.

### **scope**

The scope of the search. Can be either `one` or `sub`. Note that a scope of `base` is also supported by RFC 2255, but is not supported by this module. If the scope is not provided, or if

base scope is specified, the default is to use a scope of sub.

### **filter**

A valid LDAP search filter. If not provided, defaults to `(objectClass=*)`, which will search for all objects in the tree. Filters are limited to approximately 8000 characters (the definition of `MAX_STRING_LEN` in the Apache source code). This should be more than sufficient for any application. In 2.4.10 and later, the keyword `none` disables the use of a filter; this is required by some primitive LDAP servers.

When doing searches, the attribute, filter and username passed by the HTTP client are combined to create a search filter that looks like `(&(filter)(attribute=username))`.

For example, consider an URL of `ldap://ldap.example.com/o=Example?cn?sub?(posixid=*)`. When a client attempts to connect using a username of `Babs Jenson`, the resulting search filter will be `(&(posixid=*)(cn=Babs Jenson))`.

An optional parameter can be added to allow the LDAP Url to override the connection type. This parameter can be one of the following:

### **NONE**

Establish an unsecure connection on the default LDAP port. This is the same as `ldap://` on port 389.

### **SSL**

Establish a secure connection on the default secure LDAP port. This is the same as `ldaps://`

### **TLS | STARTTLS**

Establish an upgraded secure connection on the default LDAP port. This connection will be initiated on port 389 by

default and then upgraded to a secure connection on the same port.

See above for examples of [AuthLDAPUrl](#) URLs.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_authz_core`

<b>Description:</b>	Core Authorization
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>authz_core_module</code>
<b>Source File:</b>	<code>mod_authz_core.c</code>
<b>Compatibility:</b>	Available in Apache HTTPD 2.3 and later

### Summary

This module provides core authorization capabilities so that authenticated users can be allowed or denied access to portions of the web site. [mod\\_authz\\_core](#) provides the functionality to register various authorization providers. It is usually used in conjunction with an authentication provider module such as [mod\\_authn\\_file](#) and an authorization module such as [mod\\_authz\\_user](#). It also allows for advanced logic to be applied to the authorization processing.



Extended authorization providers can be created within the configuration file and assigned an alias name. The alias providers can then be referenced through the [Require](#) directive in the same way as a base authorization provider. Besides the ability to create and alias an extended provider, it also allows the same extended authorization provider to be referenced by multiple locations.

## Example

The example below creates two different ldap authorization provider aliases based on the ldap-group authorization provider. This example allows a single authorization location to check group membership within multiple ldap hosts:

```
<AuthzProviderAlias ldap-group ldap-group-a
  AuthLDAPBindDN cn=youruser,o=ctx
  AuthLDAPBindPassword yourpassword
  AuthLDAPURL ldap://ldap.host/o=ctx
</AuthzProviderAlias>

<AuthzProviderAlias ldap-group ldap-group-a
  AuthLDAPBindDN cn=yourotheruser,o=dev
  AuthLDAPBindPassword yourotherpassword
  AuthLDAPURL ldap://other.ldap.host/o=dev
</AuthzProviderAlias>

Alias "/secure" "/webpages/secure"
<Directory "/webpages/secure">
  Require all granted

  AuthBasicProvider file

  AuthType Basic
  AuthName LDAP_Protected_Place
```

```
#implied OR operation  
Require ldap-group-alias1  
Require ldap-group-alias2  
</Directory>
```



The authorization container directives `<RequireAll>`, `<RequireAny>` and `<RequireNone>` may be combined with each other and with the `Require` directive to express complex authorization logic.

The example below expresses the following authorization logic. In order to access the resource, the user must either be the superadmin user, or belong to both the admins group and the Administrators LDAP group and either belong to the sales group or have the LDAP dept attribute sales. Furthermore, in order to access the resource, the user must not belong to either the temps group or the LDAP group Temporary Employees.

```
<Directory "/www/mydocs">
  <RequireAll>
    <RequireAny>
      Require user superadmin
      <RequireAll>
        Require group admins
        Require ldap-group cn=Admini:
      <RequireAny>
        Require group sales
        Require ldap-attribute (
      </RequireAny>
    </RequireAll>
  </RequireAny>
  <RequireNone>
    Require group temps
    Require ldap-group cn=Temporary
  </RequireNone>
</RequireAll>
</Directory>
```



[mod\\_authz\\_core](#) provides some generic authorization providers which can be used with the [Require](#) directive.

## Require env

The `env` provider allows access to the server to be controlled based on the existence of an [environment variable](#). When `Require env env-variable` is specified, then the request is allowed access if the environment variable *env-variable* exists. The server provides the ability to set environment variables in a flexible way based on characteristics of the client request using the directives provided by [mod\\_setenvif](#). Therefore, this directive can be used to allow access based on such factors as the clients User-Agent (browser type), Referer, or other HTTP request header fields.

```
SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in
<Directory "/docroot">
    Require env let_me_in
</Directory>
```

In this case, browsers with a user-agent string beginning with `KnockKnock/2.0` will be allowed access, and all others will be denied.

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.

## Require all

The `all` provider mimics the functionality that was previously provided by the 'Allow from all' and 'Deny from all' directives. This provider can take one of two arguments which are 'granted' or 'denied'. The following examples will grant or deny access to all requests.

```
Require all granted
```

```
Require all denied
```

## Require method

The `method` provider allows using the HTTP method in authorization decisions. The GET and HEAD methods are treated as equivalent. The TRACE method is not available to this provider, use [TraceEnable](#) instead.

The following example will only allow GET, HEAD, POST, and OPTIONS requests:

```
Require method GET POST OPTIONS
```

The following example will allow GET, HEAD, POST, and OPTIONS requests without authentication, and require a valid user for all other methods:

```
<RequireAny>  
    Require method GET POST OPTIONS  
    Require valid-user  
</RequireAny>
```

## Require expr

The expr provider allows basing authorization decisions on arbitrary expressions.

```
Require expr "%{TIME_HOUR} -ge 9 && %{TIME_H
```

```
<RequireAll>  
  Require expr "!(%{QUERY_STRING} =~ /sec  
  Require expr "%{REQUEST_URI} in { '/exar  
</RequireAll>
```

```
Require expr "!(%{QUERY_STRING} =~ /secret/
```

The syntax is described in the [ap\\_expr](#) documentation.

Normally, the expression is evaluated before authentication. However, if the expression returns false and references the variable `%{REMOTE_USER}`, authentication will be performed and the expression will be re-evaluated.



## AuthMerging Directive

<b>Description:</b>	Controls the manner in which each configuration section's authorization logic is combined with that of preceding configuration sections.
<b>Syntax:</b>	AuthMerging Off   And   Or
<b>Default:</b>	AuthMerging Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_authz_core

When authorization is enabled, it is normally inherited by each subsequent [configuration section](#), unless a different set of authorization directives is specified. This is the default action, which corresponds to an explicit setting of `AuthMerging Off`.

However, there may be circumstances in which it is desirable for a configuration section's authorization to be combined with that of its predecessor while configuration sections are being merged. Two options are available for this case, `And` and `Or`.

When a configuration section contains `AuthMerging And` or `AuthMerging Or`, its authorization logic is combined with that of the nearest predecessor (according to the overall order of configuration sections) which also contains authorization logic as if the two sections were jointly contained within a [<RequireAll>](#) or [<RequireAny>](#) directive, respectively.

The setting of `AuthMerging` is not inherited outside of the configuration section in which it appears. In the following example, only users belonging to group `alpha` may access `/www/docs`. Users belonging to either groups `alpha` or `beta` may access `/www/docs/ab`. However, the default `Off` setting

of `AuthMerging` applies to the `<Directory>` configuration section for `/www/docs/ab/gamma`, so that section's authorization directives override those of the preceding sections. Thus only users belong to the group `gamma` may access `/www/docs/ab/gamma`.

```
<Directory "/www/docs">
  AuthType Basic
  AuthName Documents
  AuthBasicProvider file
  AuthUserFile "/usr/local/apache/passwd/p
  Require group alpha
</Directory>

<Directory "/www/docs/ab">
  AuthMerging Or
  Require group beta
</Directory>

<Directory "/www/docs/ab/gamma">
  Require group gamma
</Directory>
```



**Description:** Enclose a group of directives that represent an extension of a base authorization provider and referenced by the specified alias

**Syntax:** `<AuthzProviderAlias baseProvider Alias Require-Parameters> ...`  
`</AuthzProviderAlias>`

**Context:** server config

**Status:** Base

**Module:** mod\_authz\_core

`<AuthzProviderAlias>` and `</AuthzProviderAlias>` are used to enclose a group of authorization directives that can be referenced by the alias name using the directive [Require](#).



<b>Description:</b>	Send '403 FORBIDDEN' instead of '401 UNAUTHORIZED' if authentication succeeds but authorization fails
<b>Syntax:</b>	AuthzSendForbiddenOnFailure On Off
<b>Default:</b>	AuthzSendForbiddenOnFailure Off
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Base
<b>Module:</b>	mod_authz_core
<b>Compatibility:</b>	Available in Apache HTTPD 2.3.11 and later

If authentication succeeds but authorization fails, Apache HTTPD will respond with an HTTP response code of '401 UNAUTHORIZED' by default. This usually causes browsers to display the password dialogue to the user again, which is not wanted in all situations. `AuthzSendForbiddenOnFailure` allows to change the response code to '403 FORBIDDEN'.

### Security Warning

Modifying the response in case of missing authorization weakens the security of the password, because it reveals to a possible attacker, that his guessed password was right.



<b>Description:</b>	Tests whether an authenticated user is authorized by an authorization provider.
<b>Syntax:</b>	Require [not] <i>entity-name</i> [ <i>entity-name</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_authz_core

This directive tests whether an authenticated user is authorized according to a particular authorization provider and the specified restrictions. [mod\\_authz\\_core](#) provides the following generic authorization providers:

#### **Require all granted**

Access is allowed unconditionally.

#### **Require all denied**

Access is denied unconditionally.

#### **Require env *env-var* [*env-var*] ...**

Access is allowed only if one of the given environment variables is set.

#### **Require method *http-method* [*http-method*] ...**

Access is allowed only for the given HTTP methods.

#### **Require expr *expression***

Access is allowed if *expression* evaluates to true.

Some of the allowed syntaxes provided by [mod\\_authz\\_user](#), [mod\\_authz\\_host](#), and [mod\\_authz\\_groupfile](#) are:

#### **Require user *userid* [*userid*] ...**

Only the named users can access the resource.

### **Require group *group-name* [*group-name*] ...**

Only users in the named groups can access the resource.

### **Require valid-user**

All valid users can access the resource.

### **Require ip 10 172.20 192.168.2**

Clients in the specified IP address ranges can access the resource.

Other authorization modules that implement require options include [mod\\_authnz\\_ldap](#), [mod\\_authz\\_dbm](#), [mod\\_authz\\_dbd](#), [mod\\_authz\\_owner](#) and [mod\\_ssl](#).

In most cases, for a complete authentication and authorization configuration, **Require** must be accompanied by [AuthName](#), [AuthType](#) and [AuthBasicProvider](#) or [AuthDigestProvider](#) directives, and directives such as [AuthUserFile](#) and [AuthGroupFile](#) (to define users and groups) in order to work correctly. Example:

```
AuthType Basic
AuthName "Restricted Resource"
AuthBasicProvider file
AuthUserFile "/web/users"
AuthGroupFile "/web/groups"
Require group admin
```

Access controls which are applied in this way are effective for **all** methods. **This is what is normally desired.** If you wish to apply access controls only to specific methods, while leaving other methods unprotected, then place the **Require** statement into a [<Limit>](#) section.

The result of the **Require** directive may be negated through the

use of the not option. As with the other negated authorization directive `<RequireNone>`, when the `Require` directive is negated it can only fail or return a neutral result, and therefore may never independently authorize a request.

In the following example, all users in the alpha and beta groups are authorized, except for those who are also in the reject group.

```
<Directory "/www/docs">
  <RequireAll>
    Require group alpha beta
    Require not group reject
  </RequireAll>
</Directory>
```

When multiple `Require` directives are used in a single [configuration section](#) and are not contained in another authorization directive like `<RequireAll>`, they are implicitly contained within a `<RequireAny>` directive. Thus the first one to authorize a user authorizes the entire request, and subsequent `Require` directives are ignored.

### Security Warning

Exercise caution when setting authorization directives in [Location](#) sections that overlap with content served out of the filesystem. By default, these [configuration sections](#) overwrite authorization configuration in [Directory](#), and [Files](#) sections.

The [AuthMerging](#) directive can be used to control how authorization configuration sections are merged.

### See also

- [Access control howto](#)
- [Authorization Containers](#)
- [mod\\_authn\\_core](#)
- [mod\\_authz\\_host](#)



## RequireAll Directive

**Description:** Enclose a group of authorization directives of which none must fail and at least one must succeed for the enclosing directive to succeed.

**Syntax:** `<RequireAll> ... </RequireAll>`

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Base

**Module:** mod\_authz\_core

`<RequireAll>` and `</RequireAll>` are used to enclose a group of authorization directives of which none must fail and at least one must succeed in order for the `<RequireAll>` directive to succeed.

If none of the directives contained within the `<RequireAll>` directive fails, and at least one succeeds, then the `<RequireAll>` directive succeeds. If none succeed and none fail, then it returns a neutral result. In all other cases, it fails.

### See also

- [Authorization Containers](#)
- [Authentication, Authorization, and Access Control](#)



## RequireAny Directive

<b>Description:</b>	Enclose a group of authorization directives of which one must succeed for the enclosing directive to succeed.
<b>Syntax:</b>	<code>&lt;RequireAny&gt; ... &lt;/RequireAny&gt;</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_authz_core

`<RequireAny>` and `</RequireAny>` are used to enclose a group of authorization directives of which one must succeed in order for the `<RequireAny>` directive to succeed.

If one or more of the directives contained within the `<RequireAny>` directive succeed, then the `<RequireAny>` directive succeeds. If none succeed and none fail, then it returns a neutral result. In all other cases, it fails.

Because negated authorization directives are unable to return a successful result, they can not significantly influence the result of a `<RequireAny>` directive. (At most they could cause the directive to fail in the case where they failed and all other directives returned a neutral value.) Therefore negated authorization directives are not permitted within a `<RequireAny>` directive.

### See also

- [Authorization Containers](#)
- [Authentication, Authorization, and Access Control](#)



## RequireNone Directive

**Description:** Enclose a group of authorization directives of which none must succeed for the enclosing directive to not fail.

**Syntax:** `<RequireNone> ... </RequireNone>`

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Base

**Module:** mod\_authz\_core

`<RequireNone>` and `</RequireNone>` are used to enclose a group of authorization directives of which none must succeed in order for the `<RequireNone>` directive to not fail.

If one or more of the directives contained within the `<RequireNone>` directive succeed, then the `<RequireNone>` directive fails. In all other cases, it returns a neutral result. Thus as with the other negated authorization directive `Require not`, it can never independently authorize a request because it can never return a successful result. It can be used, however, to restrict the set of users who are authorized to access a resource.

Because negated authorization directives are unable to return a successful result, they can not significantly influence the result of a `<RequireNone>` directive. Therefore negated authorization directives are not permitted within a `<RequireNone>` directive.

### See also

- [Authorization Containers](#)
- [Authentication, Authorization, and Access Control](#)

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_authz_dbd`

<b>Description:</b>	Group Authorization and Login using SQL
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>authz_dbd_module</code>
<b>Source File:</b>	<code>mod_authz_dbd.c</code>
<b>Compatibility:</b>	Available in Apache 2.4 and later

## Summary

This module provides authorization capabilities so that authenticated users can be allowed or denied access to portions of the web site by group membership. Similar functionality is provided by [mod\\_authz\\_groupfile](#) and [mod\\_authz\\_dbm](#), with the exception that this module queries a SQL database to determine whether a user is a member of a group.

This module can also provide database-backed user login/logout capabilities. These are likely to be of most value when used in conjunction with [mod\\_authn\\_dbd](#).

This module relies on [mod\\_dbd](#) to specify the backend database driver and connection parameters, and manage the database connections.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Require](#)

[AuthDBDUserPWQuery](#)

[DBDriver](#)

[DBDParams](#)



Apache's [Require](#) directives are used during the authorization phase to ensure that a user is allowed to access a resource. `mod_authz_dbd` extends the authorization types with `dbd-group`, `dbd-login` and `dbd-logout`.

Since v2.4.8, [expressions](#) are supported within the DBD require directives.

## Require dbd-group

This directive specifies group membership that is required for the user to gain access.

```
Require dbd-group team
AuthzDBDQuery "SELECT group FROM authz WHERE
```

## Require dbd-login

This directive specifies a query to be run indicating the user has logged in.

```
Require dbd-login
AuthzDBDQuery "UPDATE authn SET login = 'tru
```

## Require dbd-logout

This directive specifies a query to be run indicating the user has logged out.

```
Require dbd-logout
AuthzDBDQuery "UPDATE authn SET login = 'fa
```



## Database Login

In addition to the standard authorization function of checking group membership, this module can also provide server-side user session management via database-backed login/logout capabilities. Specifically, it can update a user's session status in the database whenever the user visits designated URLs (subject of course to users supplying the necessary credentials).

This works by defining two special [Require](#) types: `Require dbd-login` and `Require dbd-logout`. For usage details, see the configuration example below.



## Client-Side Integration

Some administrators may wish to implement client-side session management that works in concert with the server-side login/logout capabilities offered by this module, for example, by setting or unsetting an HTTP cookie or other such token when a user logs in or out.

To support such integration, `mod_authz_dbd` exports an optional hook that will be run whenever a user's status is updated in the database. Other session management modules can then use the hook to implement functions that start and end client-side sessions.



```
# mod_dbd configuration
DBDriver pgsq1
DBDParams "dbname=apacheauth user=apache pas

DBDMin 4
DBDKeep 8
DBDMax 20
DBDExptime 300

<Directory "/usr/www/my.site/team-private/">
  # mod_authn_core and mod_auth_basic config
  # for mod_authn_dbd
  AuthType Basic
  AuthName Team
  AuthBasicProvider dbd

  # mod_authn_dbd SQL query to authenticate
  AuthDBDUserPWQuery \
    "SELECT password FROM authn WHERE user =

  # mod_authz_core configuration for mod_authn_dbd
  Require dbd-group team

  # mod_authz_dbd configuration
  AuthzDBDQuery "SELECT group FROM authz WHERE

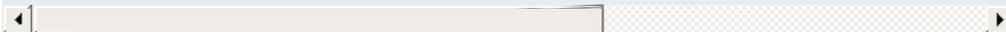
  # when a user fails to be authenticated on this page
  # invite them to login; this page should point
  # to /team-private/login.html
  ErrorDocument 401 "/login-info.html"

<Files "login.html">
  # don't require user to already be logged in
  AuthDBDUserPWQuery "SELECT password FROM
```

```
# dbd-login action executes a statement
Require dbd-login
AuthzDBDQuery "UPDATE authn SET login =

# return user to referring page (if any)
# successful login
AuthzDBDLoginToReferer On
</Files>

<Files "logout.html">
  # dbd-logout action executes a statement
  Require dbd-logout
  AuthzDBDQuery "UPDATE authn SET login =
</Files>
</Directory>
```



**Description:** Determines whether to redirect the Client to the Referring page on successful login or logout if a Referrer request header is present

**Syntax:** AuthzDBDLoginToReferer On|Off

**Default:** AuthzDBDLoginToReferer Off

**Context:** directory

**Status:** Extension

**Module:** mod\_authz\_dbd

In conjunction with `Require dbd-login` or `Require dbd-logout`, this provides the option to redirect the client back to the Referring page (the URL in the Referrer HTTP request header, if present). When there is no Referrer header, `AuthzDBDLoginToReferer On` will be ignored.



<b>Description:</b>	Specify the SQL Query for the required operation
<b>Syntax:</b>	<code>AuthzDBDQuery query</code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_authz_dbd</code>

The `AuthzDBDQuery` specifies an SQL query to run. The purpose of the query depends on the `Require` directive in effect.

- When used with a `Require dbd-group` directive, it specifies a query to look up groups for the current user. This is the standard functionality of other authorization modules such as `mod_authz_groupfile` and `mod_authz_dbm`. The first column value of each row returned by the query statement should be a string containing a group name. Zero, one, or more rows may be returned.

```
Require dbd-group
AuthzDBDQuery "SELECT group FROM groups"
```

- When used with a `Require dbd-login` or `Require dbd-logout` directive, it will never deny access, but will instead execute a SQL statement designed to log the user in or out. The user must already be authenticated with `mod_authn_dbd`.

```
Require dbd-login
AuthzDBDQuery "UPDATE authn SET login ="
```

In all cases, the user's ID will be passed as a single string

parameter when the SQL query is executed. It may be referenced within the query statement using a %s format specifier.



<b>Description:</b>	Specify a query to look up a login page for the user
<b>Syntax:</b>	<code>AuthzDBDRedirectQuery <i>query</i></code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_authz_dbd</code>

Specifies an optional SQL query to use after successful login (or logout) to redirect the user to a URL, which may be specific to the user. The user's ID will be passed as a single string parameter when the SQL query is executed. It may be referenced within the query statement using a %s format specifier.

```
AuthzDBDRedirectQuery "SELECT userpage FROM
```

The first column value of the first row returned by the query statement should be a string containing a URL to which to redirect the client. Subsequent rows will be ignored. If no rows are returned, the client will not be redirected.

Note that `AuthzDBDLoginToReferrer` takes precedence if both are set.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_authz\_dbm

- DBM
- Extension
- authz\_dbm\_module
- mod\_authz\_dbm.c
- 2.1

[mod\\_authz\\_groupfile](#) .



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[Require](#)

[Satisfy](#)



## AuthDBMGroupFile

```
AuthDBMGroupFile file-path
directory, .htaccess
Override : AuthConfig
Extension
mod_authz_dbm
```

AuthDBMGroupFile DBM .  
path .

```
AuthDBMGroupFile .
AuthDBMGroupFile .
```

DBM DBM : DBM .  
DBM :

```
AuthDBMGroupFile /www/userbase
AuthDBMUserFile /www/userbase
```

DBM .

```
[ : ( ) ]
```



## AuthzDBMType

```
AuthzDBMType default|SDBM|GDBM|NDBM|DB
AuthzDBMType default
directory, .htaccess
Override : AuthConfig
Extension
mod_authz_dbm
```





| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_auth\_groupfile

- ┆
- ┆ Base
- ┆ auth\_groupfile\_module
- ┆ mod\_auth\_groupfile.c
- ┆ 2.1



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[Require](#)

[Satisfy](#)



## AuthGroupFile

```
:  
: AuthGroupFile file-path  
: directory, .htaccess  
Override : AuthConfig  
: Base  
: mod_authz_groupfile
```

### AuthGroupFile

```
. ServerRoot .  
'  
'
```

```
:  
mygroup: bob joe anne
```

```
. AuthDBMGroupFile  
.
```

```
AuthGroupFile .  
, AuthGroupFile .
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_authz_host`

<b>Description:</b>	Group authorizations based on host (name or IP address)
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>authz_host_module</code>
<b>Source File:</b>	<code>mod_authz_host.c</code>
<b>Compatibility:</b>	The forward-dns provider was added in 2.4.19

## Summary

The authorization providers implemented by `mod_authz_host` are registered using the `Require` directive. The directive can be referenced within a `<Directory>`, `<Files>`, or `<Location>` section as well as `.htaccess` files to control access to particular parts of the server. Access can be controlled based on the client hostname or IP address.

In general, access restriction directives apply to all access methods (GET, PUT, POST, etc). This is the desired behavior in most cases. However, it is possible to restrict some methods, while leaving other methods unrestricted, by enclosing the directives in a `<Limit>` section.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Authentication, Authorization, and Access Control Require](#)



## THE REQUIRE DIRECTIVE

Apache's [Require](#) directive is used during the authorization phase to ensure that a user is allowed or denied access to a resource. `mod_authz_host` extends the authorization types with `ip`, `host`, `forward-dns` and `local`. Other authorization types may also be used but may require that additional authorization modules be loaded.

These authorization providers affect which hosts can access an area of the server. Access can be controlled by hostname, IP Address, or IP Address range.

Since v2.4.8, [expressions](#) are supported within the host require directives.

### Require ip

The `ip` provider allows access to the server to be controlled based on the IP address of the remote client. When `Require ip ip-address` is specified, then the request is allowed access if the IP address matches.

A full IP address:

```
Require ip 10.1.2.3
Require ip 192.168.1.104 192.168.1.205
```

An IP address of a host allowed access

A partial IP address:

```
Require ip 10.1
Require ip 10 172.20 192.168.2
```

The first 1 to 3 bytes of an IP address, for subnet restriction.

A network/netmask pair:

```
Require ip 10.1.0.0/255.255.0.0
```

A network a.b.c.d, and a netmask w.x.y.z. For more fine-grained subnet restriction.

A network/nnn CIDR specification:

```
Require ip 10.1.0.0/16
```

Similar to the previous case, except the netmask consists of nnn high-order 1 bits.

Note that the last three examples above match exactly the same set of hosts.

IPv6 addresses and IPv6 subnets can be specified as shown below:

```
Require ip 2001:db8::a00:20ff:fea7:ccea  
Require ip 2001:db8:1:1::a  
Require ip 2001:db8:2:1::/64  
Require ip 2001:db8:3::/48
```

Note: As the IP addresses are parsed on startup, expressions are not evaluated at request time.

## Require host

The host provider allows access to the server to be controlled based on the host name of the remote client. When `Require host host-name` is specified, then the request is allowed access if the host name matches.

A (partial) domain-name

```
Require host example.org  
Require host .net example.edu
```

Hosts whose names match, or end in, this string are allowed access. Only complete components are matched, so the above example will match `foo.example.org` but it will not match `fooexample.org`. This configuration will cause Apache to perform a double reverse DNS lookup on the client IP address, regardless of the setting of the [HostnameLookups](#) directive. It will do a reverse DNS lookup on the IP address to find the associated hostname, and then do a forward lookup on the hostname to assure that it matches the original IP address. Only if the forward and reverse DNS are consistent and the hostname matches will access be allowed.

## Require forward-dns

The `forward-dns` provider allows access to the server to be controlled based on simple host names. When `Require forward-dns host-name` is specified, all IP addresses corresponding to *host-name* are allowed access.

In contrast to the `host` provider, this provider does not rely on reverse DNS lookups: it simply queries the DNS for the host name and allows a client if its IP matches. As a consequence, it will only work with host names, not domain names. However, as the reverse DNS is not used, it will work with clients which use a dynamic DNS service.

```
Require forward-dns bla.example.org
```

A client the IP of which is resolved from the name

`bla.example.org` will be granted access.

The `forward-dns` provider was added in 2.4.19.

## Require local

The `local` provider allows access to the server if any of the following conditions is true:

- the client address matches `127.0.0.0/8`
- the client address is `::1`
- both the client and the server address of the connection are the same

This allows a convenient way to match connections that originate from the local host:

```
Require local
```

## Security Note

If you are proxying content to your server, you need to be aware that the client address will be the address of your proxy server, not the address of the client, and so using the `Require` directive in this context may not do what you mean. See [mod\\_remoteip](#) for one possible solution to this problem.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_authz\_owner

- Extension
- authz\_owner\_module
- mod\_authz\_owner.c
- 2.1

```
HTTP ( /
      mod_auth_basic mod_auth_digest
mod_authz_owner Require , file-owner
file-group:
file-owner
      jones.
file-group
      mod_authz_groupfile mod_authz_d
      accounts
      .
```

```
mod_authz_owner ( , ),
      "MultiViews" .
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[Require](#)

[Satisfy](#)



## Require file-owner

```
                                ~/public_html/p
                                AuthDBMUserFile ,
smith                            /home/smith/public_html/private
```

```
<Directory /home/*/public_html/private>
  AuthType Basic
  AuthName MyPrivateFiles
  AuthBasicProvider dbm
  AuthDBMUserFile /usr/local/apache2/etc/.htdbm-all
  Satisfy All
  Require file-owner
</Directory>
```

## Require file-group

```
                                ~/public_html/project-foo
                                foo ,
                                AuthDBM
,   foo   .   jones smith   foo ,
project-foo   .
```

```
<Directory /home/*/public_html/project-foo>
  AuthType Basic
  AuthName "Project Foo Files"
  AuthBasicProvider dbm

  # combined user/group database
  AuthDBMUserFile /usr/local/apache2/etc/.htdbm-all
  AuthDBMGroupFile /usr/local/apache2/etc/.htdbm-all

  Satisfy All
  Require file-group
</Directory>
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_authz\_user

- ┆
- ┆ Base
- ┆ authz\_user\_module
- ┆ mod\_authz\_user.c
- ┆ 2.1

Require user  
user

, re



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[Require](#)

[Satisfy](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_autoindex

```
ls Win32 dir
```

```
Base
```

```
autoindex_module
```

```
mod_autoindex.c
```

```
:
```

```
• index.html . Direct
```

```
• mod\_dir .
```

```
• AddIcon
```

```
AddIconByEncoding, AddIconByType .
```

```
mod\_autoindex .
```

```
, () .
```

```
Options +Indexes . Options .
```

```
IndexOptions FancyIndexing , .
```

```
IndexOptions SuppressColumnSorting
```

```
"Size()" ., 1010 1011
```

```
"1K" 1010 .
```



## 2.0.23 , [IndexOptions IgnoreClient](#) .

- C=N
- C=M ,
- C=S ,
- C=D ,
  
- O=A
- O=D
  
- F=0 (FancyIndexed )
- F=1 FancyIndexed
- F=2 HTMLTable FancyIndexed
  
- V=0
- V=1
  
- P=*pattern* *pattern*

'P'attern [IndexIgnore](#) , autoindex  
[mod\\_autoindex](#) .

header.html . submit  
"X" mod\_autoindex X=Go .

```
<form action="" method="get">
  Show me a <select name="F">
    <option value="0"> Plain list</option>
    <option value="1" selected="selected"> Fancy list</option>
    <option value="2"> Table list</option>
  </select>
```

```
Sorted by <select name="C">
  <option value="N" selected="selected"> Name</option>
  <option value="M"> Date Modified</option>
  <option value="S"> Size</option>
  <option value="D"> Description</option>
</select>
<select name="O">
  <option value="A" selected="selected"> Ascending</option>
  <option value="D"> Descending</option>
</select>
<select name="V">
  <option value="0" selected="selected"> in Normal
order</option>
  <option value="1"> in Version order</option>
</select>
Matching <input type="text" name="P" value="*" />
<input type="submit" name="X" value="Go" />
</form>
```



## APACHE

```
:  
:  
: AddAlt string file [file] ...  
: , , directory, .htaccess  
Override : Indexes  
:  
: Base  
:  
: mod_autoindex
```

```
AddAlt FancyIndexing . F  
, , , . String  
· , , ·
```

```
AddAlt "PDF file" *.pdf  
AddAlt Compressed *.gz *.zip *.Z
```



## Adding Encoding

```

: MIME-encoding
: AddAltByEncoding string MIME-encoding
  [MIME-encoding] ...
: , , directory, .htaccess
Override : Indexes
: Base
: mod_autoindex

```

```

AddAltByEncoding FancyIndexing .
MIME-encoding x-compress content-encoding. Str
( " ' ) . , ,
.

```

```

AddAltByEncoding gzip x-gzip

```



## AddAltByType

```
:_ MIME content-type
:_ AddAltByType string MIME-type [MIME-
  type] ...
:_ , , directory, .htaccess
Override : Indexes
:_ Base
:_ mod_autoindex
```

```
AddAltByType FancyIndexing .
type text/html content-type. String ( "
') . , ,
.
```

```
AddAltByType 'plain text' text/plain
```



## AddDescription

```
:  
: AddDescription string file [file] ...  
: , , directory, .htaccess  
Override : Indexes  
: Base  
: mod_autoindex
```

```
FancyIndexing . File ,  
String ( " ) .
```

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

```
23. IndexOptions SuppressIcon  
6 , IndexOptions SuppressSize 7 ,  
IndexOptions SuppressLastModified 19 .  
55 .
```

```
DescriptionWidth Inde
```

```
AddDescription character entity ( ; &lt; , & ;  
) HTML . ( )  
) .
```



## ADDICON

```
:  
: AddIcon icon name [name] ...  
: , , directory, .htaccess  
Override : Indexes  
: Base  
: mod_autoindex
```

FancyIndexing *name* .  
escaped) URL (*alttext, url*) . *alttext*

*Name* ^^DIRECTORY^^, ( )  
^^BLANKICON^^, , , .

```
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm  
AddIcon /icons/dir.xbm ^^DIRECTORY^^  
AddIcon /icons/backup.xbm *~
```

AddIcon AddIconByType .



## Adding Encoding

```

: MIME content-encoding
: AddIconByEncoding icon MIME-encoding
[MIME-encoding] ...
: , , directory, .htaccess
Override : Indexes
: Base
: mod_autoindex

```

```

FancyIndexing . Icon (%-esc:
URL (alttext,url) . alttext
.
MIME-encoding content-encoding .

```

```
AddIconByEncoding /icons/compress.xbm x-compress
```



## AddIconByType

```
:_ MIME content-type
:_ AddIconByType icon MIME-type [MIME-
type] ...
:_ , , directory, .htaccess
Override : Indexes
:_ Base
:_ mod_autoindex
```

```
FancyIndexing MIME-type .
(%-escaped) URL (alttext,url) . alt
```

MIME-type mime type .

```
AddIconByType (IMG,/icons/image.xbm) image/*
```



## DefaultIcon

```
DefaultIcon url-path
, , directory, .htaccess
Override : Indexes
Base
mod_autoindex
```

DefaultIcon [FancyIndexing](#) .  
Icon (%-escaped) URL.

```
DefaultIcon /icon/unknown.xbm
```



## HeaderName

```
:  
: HeaderName filename  
: , , directory,  
: .htaccess  
Override : Indexes  
: Base  
: mod_autoindex
```

HeaderName . Filename

```
HeaderName HEADER.html
```

```
HeaderName ReadmeName Filename  
URI . Filename DocumentRoot  
  
HeaderName /include/HEADER.html  
  
Filename major content type text/* ( , text/html,  
text/plain, ) . , (  
text/html filename CGI :  
  
AddType text/html .cgi  
  
Options MultiViews . filename (CGI )  
text/html options Includes IncludesNOEXEC  
server-side includes . ( mod_include )
```

HeaderName (<html>, <head>, ) HTML

[IndexOptions +SuppressHTMLPreamble](#)

.



- [:](#) Inserts text in the HEAD section of an index page.
- [:](#)
- [:](#) , , directory, .htaccess
- [:](#) Base
- [:](#) mod\_autoindex

Documentation not yet translated. Please see English version of document.



## IndexIgnore

```
IndexIgnore file [file] ...  
            , , directory, .htaccess  
Override : Indexes  
            Base  
            mod_autoindex
```

**IndexIgnore** . *File*  
 . IndexIgnore .  
 . ( ) .

```
IndexIgnore README .htaccess *.bak *~
```



<a href="#">:</a>	Empties the list of files to hide when listing a directory
<a href="#">:</a>	IndexIgnoreReset ON OFF
<a href="#">:</a>	, , directory, .htaccess
<b><a href="#">Override :</a></b>	Indexes
<a href="#">:</a>	Base
<a href="#">:</a>	mod_autoindex
<a href="#">:</a>	2.3.10 and later

The documentation for this directive has not been translated yet. Please have a look at the English version.



## IndexOptions

```
IndexOptions [+|-]option [[+|-]option]
...
, , directory, .htaccess
Override : Indexes
Base
mod_autoindex
```

### IndexOptions . Option

**DescriptionWidth=[n | \*] ( 2.0.23 )**  
DescriptionWidth .  
-DescriptionWidth ( ) [mod\\_](#)  
DescriptionWidth=n n .  
DescriptionWidth=\* .  
**AddDescription** .

### FancyIndexing

fancy .

### FoldersFirst ( 2.0.23 )

, .  
,  
FoldersFirst Zed Beta ,  
Gamma Alpha . **FancyIndexing**

### HTMLTable (, 2.0.23 )

FancyIndexing HTML fancy .  
. WinNT  
( ) .

### IconsAreLinks

fancy .

### IconHeight[=*pixels*]

IconWidth img height  
width . .

### IconWidth[=*pixels*]

IconHeight img height  
width . .

### IgnoreCase

, IgnoreCase  
Zeta alfa (: GAMMA gamma  
).

### IgnoreClient

[mod\\_autoindex](#) .  
([SuppressColumnSorting](#).)

### NameWidth=[*n* | \*]

NameWidth .  
-NameWidth ( [mod\\_autoindex](#)  
NameWidth=*n* *n* .  
NameWidth=\* .

### ScanHTMLTitles

fancy HTML title . [AddDescription](#)  
title . CPU .

### SuppressColumnSorting

FancyIndexed .  
,  
2.0.23 [IndexOptions](#)

### SuppressDescription

fancy . , 23

[AddDescription](#) .

[DescriptionWidth](#) .

### SuppressHTMLPreamble

[HeaderName](#) HTML

(<html>, <head>, *et cetera*) .

SuppressHTMLPreamble header .

header HTML .header

### SuppressIcon ( 2.0.23 )

fancy . SuppressIcon SuppressRules

, (FancyIndexed ) pre img hr

HTML 3.2 .

### SuppressLastModified

fancy .

### SuppressRules ( 2.0.23 )

( hr ) SuppressIcon SuppressRule

, (FancyIndexed ) pre img hr

HTML 3.2 .

### SuppressSize

fancy .

### TrackModified ( 2.0.23 )

HTTP Last-Modified ETag .

stat() . OS2 JFS, Win3:

. , OS2 Win32 FAT .

Modified .

### VersionSort ( 2.0a3 )

VersionSort .

```
:
foo-1.7
foo-1.7.2
foo-1.7.12
foo-1.8.2
foo-1.8.2a
foo-1.12
```

0, :

```
foo-1.001
foo-1.002
foo-1.030
foo-1.04
```

## XHTML ( 2.0.49 )

XHTML

[mod\\_autoindex](#) HTML 3.2 XHTML 1.0

.

## IndexOptions

1.3.3 IndexOptions .:

- IndexOptions .

```
<Directory /foo>
  IndexOptions HTMLTable
  IndexOptions SuppressColumnsorting
</Directory>
```

```
IndexOptions HTMLTable SuppressColumnsorting
```

- ( , + - ) .

'+' '-' ( )

.

.

:

```
IndexOptions +ScanHTMLTitles -IconsAreLinks FancyIndexing  
IndexOptions +SuppressSize
```

```
FancyIndexing  
IndexOptions FancyIndexing +SuppressSize.
```

```
IndexOptions +
```



## INDEXORDERDEFAULT

```
IndexOrderDefault Ascending|Descending  
Name|Date|Size|Description  
IndexOrderDefault Ascending Name  
, , directory, .htaccess  
Override : Indexes  
Base  
mod_autoindex
```

```
IndexOrderDefault FancyIndexing .  
fancyindexed . IndexOrderDefault
```

```
IndexOrderDefault .  
) Descending () .  
Date, Size, Description .
```

### SuppressColumnSorting



## IndexStyleSheet

<a href="#">:</a>	CSS
<a href="#">:</a>	IndexStyleSheet <i>url-path</i>
<a href="#">:</a>	, , directory, .htaccess
<a href="#">Override :</a>	Indexes
<a href="#">:</a>	Base
<a href="#">:</a>	mod_autoindex

IndexStyleSheet CSS .

### Example

```
IndexStyleSheet "/css/style.css"
```



## ReadmeName

```
ReadmeName filename  
, , directory,  
.htaccess  
Override : Indexes  
Base  
mod_autoindex
```

ReadmeName . Filename  
. Filename DocumentRoot .

```
ReadmeName FOOTER.html
```

```
2  
ReadmeName /include/FOOTER.html
```

### HeaderName .



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_brotli

<b>Description:</b>	Compress content via Brotli before it is delivered to the client
<b>Status:</b>	Extension
<b>Module Identifier:</b>	brotli_module
<b>Source File:</b>	mod_brotli.c
<b>Compatibility:</b>	Available in version 2.4.26 and later.

## Summary

The `mod_brotli` module provides the `BROTLI_COMPRESS` output filter that allows output from your server to be compressed using the brotli compression format before being sent to the client over the network. This module uses the Brotli library found at <https://github.com/google/brotli>.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Filters](#)



### Compression and TLS

Some web applications are vulnerable to an information disclosure attack when a TLS connection carries compressed data. For more information, review the details of the "BREACH" family of attacks.

This is a simple configuration that compresses common text-based content types.

### Compress only a few types

```
AddOutputFilterByType BROTLI_COMPRESS text/html text/plain text,
```



## Compression and TLS

Some web applications are vulnerable to an information disclosure attack when a TLS connection carries compressed data. For more information, review the details of the "BREACH" family of attacks.

## Output Compression

Compression is implemented by the `BROTLI_COMPRESS` [filter](#). The following directive will enable compression for documents in the container where it is placed:

```
SetOutputFilter BROTLI_COMPRESS
SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|pr
```

If you want to restrict the compression to particular MIME types in general, you may use the [AddOutputFilterByType](#) directive. Here is an example of enabling compression only for the html files of the Apache documentation:

```
<Directory "/your-server-root/manual">
    AddOutputFilterByType BROTLI_COMPRESS te
</Directory>
```

### Note

The `BROTLI_COMPRESS` filter is always inserted after `RESOURCE` filters like PHP or SSI. It never touches internal subrequests.

### Note

There is an environment variable `no-brotli`, set via [SetEnv](#), which will disable brotli compression for a particular request, even if it is supported by the client.



## Working with proxy servers

The `mod_brotli` module sends a `Vary: Accept-Encoding` HTTP response header to alert proxies that a cached response should be sent only to clients that send the appropriate `Accept-Encoding` request header. This prevents compressed content from being sent to a client that will not understand it.

If you use some special exclusions dependent on, for example, the `User-Agent` header, you must manually configure an addition to the `Vary` header to alert proxies of the additional restrictions. For example, in a typical configuration where the addition of the `BROTLI_COMPRESS` filter depends on the `User-Agent`, you should add:

```
Header append Vary User-Agent
```

If your decision about compression depends on other information than request headers (e.g. HTTP version), you have to set the `Vary` header to the value `*`. This prevents compliant proxies from caching entirely.

### Example

```
Header set Vary *
```



Since `mod_brotli` re-compresses content each time a request is made, some performance benefit can be derived by pre-compressing the content and telling `mod_brotli` to serve them without re-compressing them. This may be accomplished using a configuration like the following:

```
<IfModule mod_headers.c>
  # Serve brotli compressed CSS files if the client accepts brotli
  # and the client accepts brotli.
  RewriteCond "%{HTTP:Accept-encoding}" "br"
  RewriteCond "%{REQUEST_FILENAME}\.br" ".br"
  RewriteRule "^(.*)\.css" "$1$2" [R=301,L]

  # Serve brotli compressed JS files if the client accepts brotli
  # and the client accepts brotli.
  RewriteCond "%{HTTP:Accept-encoding}" "br"
  RewriteCond "%{REQUEST_FILENAME}\.br" ".br"
  RewriteRule "^(.*)\.js" "$1$2" [R=301,L]

  # Serve correct content types, and prevent double encoding
  RewriteRule "\.css\.br$" "-" [T=text/css]
  RewriteRule "\.js\.br$" "-" [T=text/javascript]

  <FilesMatch "(\.js\.br|\.css\.br)$">
    # Serve correct encoding type.
    Header append Content-Encoding br

    # Force proxies to cache brotli & non-brotli css/js files separately.
    Header append Vary Accept-Encoding
  </FilesMatch>
</IfModule>
```



## BrotliAlterETag Directive

<b>Description:</b>	How the outgoing ETag header should be modified during compression
<b>Syntax:</b>	BrotliAlterETag AddSuffix   NoChange   Remove
<b>Default:</b>	BrotliAlterETag AddSuffix
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_brotli

The `BrotliAlterETag` directive specifies how the ETag header should be altered when a response is compressed.

### AddSuffix

Append the compression method onto the end of the ETag, causing compressed and uncompressed representations to have unique ETags. In another dynamic compression module, `mod_deflate`, this has been the default since 2.4.0. This setting prevents serving "HTTP Not Modified" (304) responses to conditional requests for compressed content.

### NoChange

Don't change the ETag on a compressed response. In another dynamic compression module, `mod_deflate`, this has been the default prior to 2.4.0. This setting does not satisfy the HTTP/1.1 property that all representations of the same resource have unique ETags.

### Remove

Remove the ETag header from compressed responses. This prevents some conditional requests from being possible, but avoids the shortcomings of the preceding options.



<b>Description:</b>	Maximum input block size
<b>Syntax:</b>	<code>BrotliCompressionMaxInputBlock</code> <i>value</i>
<b>Default:</b>	(automatic)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_brotli

The `BrotliCompressionMaxInputBlock` directive specifies the maximum input block size between 16 and 24, with the caveat that larger block sizes require more memory.



## BrotliCompressionQuality Directive

<b>Description:</b>	Compression quality
<b>Syntax:</b>	<code>BrotliCompressionQuality <i>value</i></code>
<b>Default:</b>	<code>BrotliCompressionQuality 5</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_brotli</code>

The `BrotliCompressionQuality` directive specifies the compression quality (a value between 0 and 11). Higher quality values result in better, but also slower compression.



<b>Description:</b>	Brotli sliding compression window size
<b>Syntax:</b>	<code>BrotliCompressionWindow</code> <i>value</i>
<b>Default:</b>	<code>BrotliCompressionWindow</code> 18
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_brotli</code>

The `BrotliCompressionWindow` directive specifies the brotli sliding compression window size (a value between 10 and 24). Larger window sizes can improve compression quality, but require more memory.



<b>Description:</b>	Places the compression ratio in a note for logging
<b>Syntax:</b>	<code>BrotliFilterNote [type] notename</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_brotli</code>

The `BrotliFilterNote` directive specifies that a note about compression ratios should be attached to the request. The name of the note is the value specified for the directive. You can use that note for statistical purposes by adding the value to your [access log](#).

### Example

```
BrotliFilterNote ratio

LogFormat "%r" %b (%{ratio}n) "%{User-agent}i" brotli
CustomLog "logs/brotli_log" brotli
```

If you want to extract more accurate values from your logs, you can use the `type` argument to specify the type of data left as a note for logging. `type` can be one of:

### Input

Store the byte count of the filter's input stream in the note.

### Output

Store the byte count of the filter's output stream in the note.

### Ratio

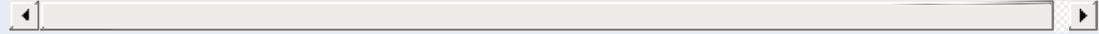
Store the compression ratio (`output/input * 100`) in the note. This is the default, if the `type` argument is omitted.

Thus you may log it this way:

### Accurate Logging

```
BrotliFilterNote Input instream
BrotliFilterNote Output outstream
BrotliFilterNote Ratio ratio
```

```
LogFormat '"%r" %{outstream}n/%{instream}n (%{ratio}n%)' brotli:
CustomLog "logs/brotli_log" brotli
```



## See also

- [mod\\_log\\_config](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_buffer

<b>Description:</b>	Support for request buffering
<b>Status:</b>	Extension
<b>Module Identifier:</b>	buffer_module
<b>Source File:</b>	mod_buffer.c
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

This module provides the ability to buffer the input and output filter stacks.

Under certain circumstances, content generators might create content in small chunks. In order to promote memory reuse, in memory chunks are always 8k in size, regardless of the size of the chunk itself. When many small chunks are generated by a request, this can create a large memory footprint while the request is being processed, and an unnecessarily large amount of data on the wire. The addition of a buffer collapses the response into the fewest chunks possible.

When httpd is used in front of an expensive content generator, buffering the response may allow the backend to complete processing and release resources sooner, depending on how the backend is designed.

The buffer filter may be added to either the input or the output filter stacks, as appropriate, using the [SetInputFilter](#), [SetOutputFilter](#), [AddOutputFilter](#) or [AddOutputFilterByType](#) directives.

### Using buffer with mod\_include

```
AddOutputFilterByType INCLUDES;BUFFER text/html
```

---

The buffer filters read the request/response into RAM and then repack the request/response into the fewest memory buckets possible, at the cost of CPU time. When the request/response is already efficiently packed, buffering the request/response could cause the request/response to be slower than not using a buffer at all. These filters should be used with care, and only where necessary.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Filters](#)



<b>Description:</b>	Maximum size in bytes to buffer by the buffer filter
<b>Syntax:</b>	BufferSize integer
<b>Default:</b>	BufferSize 131072
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_buffer

The **BufferSize** directive specifies the amount of data in bytes that will be buffered before being read from or written to each request. The default is 128 kilobytes.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_cache

- URI
- Experimental
- cache\_module
- mod\_cache.c

...

mod\_cache

[RFC 261](#)

mod\_cache (storage management module)

:

mod\_cache\_disk

mod\_mem\_cache

mod\_mem\_cache

*proxy*) [ProxyPass](#) mod\_mem\_cache , (mod\_proxy

URI



<u>mod cache disk</u>	<u>CacheRoot</u>
<u>mod mem cache</u>	<u>CacheSize</u>
	<u>CacheGcInterval</u>
	<u>CacheDirLevels</u>
	<u>CacheDirLength</u>
	<u>CacheExpiryCheck</u>
	<u>CacheMinFileSize</u>
	<u>CacheMaxFileSize</u>
	<u>CacheTimeMargin</u>
	<u>CacheGcDaily</u>
	<u>CacheGcUnused</u>
	<u>CacheGcClean</u>
	<u>CacheGcMemUsage</u>
	<u>MCacheSize</u>
	<u>MCacheMaxObjectCount</u>
	<u>MCacheMinObjectSize</u>
	<u>MCacheMaxObjectSize</u>
	<u>MCacheRemovalAlgorithm</u>
	<u>MCacheMaxStreamingBuffer</u>



## Sample httpd.conf

```
#
#
#
LoadModule cache_module modules/mod_cache.so

<IfModule mod_cache.c>
  #LoadModule cache_disk_module modules/mod_cache_disk.so
  <IfModule mod_cache_disk.c>
    CacheRoot c:/cacheroot
    CacheSize 256
    CacheEnable disk /
    CacheDirLevels 5
    CacheDirLength 3
  </IfModule>

  LoadModule mem_cache_module modules/mod_mem_cache.so
  <IfModule mod_mem_cache.c>
    CacheEnable mem /
    MCacheSize 4096
    MCacheMaxObjectCount 100
    MCacheMinObjectSize 1
    MCacheMaxObjectSize 2048
  </IfModule>
</IfModule>
```



## CacheDefaultExpire

```
CacheDefaultExpire seconds
CacheDefaultExpire 3600 (one hour)
Experimental
mod_cache
```

CacheDefaultExpire

CacheMaxExpire

```
CacheDefaultExpire 86400
```



## Cache-Details-Header

- [:](#) Add an X-Cache-Detail header to the response.
- [:](#)
- [:](#) , , directory, .htaccess
- [:](#) Experimental
- [:](#) mod\_cache

Documentation not yet translated. Please see English version of document.



## CacheDisable

```
URL  
CacheDisable url-string  
,  
Experimental  
mod_cache
```

CacheDisable mod\_cache *url-string* url

```
CacheDisable /local_files
```



## CacheEnable

```
URL  
CacheEnable cache_type url-string  
,  
Experimental  
mod_cache
```

```
CacheEnable mod_cache url-string url .  
cache_type . cache_type mem mod_mem_cache  
. cache_type disk mod_cache disk .  
cache_type fd mod_mem_cache .
```

```
( ) URL CacheEnable  
CacheEnable .
```

```
CacheEnable mem /manual  
CacheEnable fd /images  
CacheEnable disk /
```



## Cache-Header

- [: Add an X-Cache header to the response.](#)
- [:](#)
- [: , , directory, .htaccess](#)
- [: Experimental](#)
- [: mod\\_cache](#)

Documentation not yet translated. Please see English version of document.



CacheIgnoreCacheControl

```
CacheIgnoreCacheControl On|Off  
CacheIgnoreCacheControl Off  
,  
Experimental  
mod_cache
```

no-cache no-store  
CacheIgnoreCacheControl .  
CacheIgnoreCacheControl On no-cache no-store

CacheIgnoreCacheControl On



## CacheIgnoreHeaders

```
[: HTTP ()
[: CacheIgnoreHeaders header-string [header-string] ...
[: CacheIgnoreHeaders None
[: ,
[: Experimental
[: mod_cache
```

RFC 2616 (hop-by-hop) HTTP . HTTP  
, **CacheIgnoreHeaders** .

- Connection
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade

**CacheIgnoreHeaders** HTTP . ,  
(cookie) .

**CacheIgnoreHeaders** HTTP .  
(RFC 2616) , **CacheIgnoreHeaders**  
.

**1**  
CacheIgnoreHeaders Set-Cookie

**2**  
CacheIgnoreHeaders None

```
:  
CacheIgnoreHeaders Expires  
, mod_cache .
```



CacheIgnoreNoLastMod

```
: Last Modified .  
: CacheIgnoreNoLastMod On|Off  
: CacheIgnoreNoLastMod Off  
:  
: Experimental  
: mod_cache
```

```
. ( mo  
. CacheIgnoreNoLastMod  
. CacheDefaultExpire .
```

```
CacheIgnoreNoLastMod On
```



- Ignore query string when caching
- 
- ,
- Experimental
- mod\_cache

Documentation not yet translated. Please see English version of document.



- Ignore defined session identifiers encoded in the URL when caching
- 
- ,
- Experimental
- mod\_cache

Documentation not yet translated. Please see English version of document.



- Override the base URL of reverse proxied cache keys.
- 
- ,
- Experimental
- mod\_cache

Documentation not yet translated. Please see English version of document.



```
┆ LastModified .
┆ CacheLastModifiedFactor float
┆ CacheLastModifiedFactor 0.1
┆ ,
┆ Experimental
┆ mod_cache
```

### CacheLastModifiedFactor

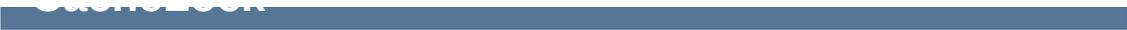
expiry-period = time-since-last-modified-date \*  
factor expiry-date = current-date + expiry-period  
, 10 factor 0.1 10\*01 = 1 .

3:00pm 3:00pm + 1 = 4:00pm.

CacheMaxExpire .

```
CacheLastModifiedFactor 0.5
```





- Enable the thundering herd lock.
- 
- ,
- Experimental
- mod\_cache

Documentation not yet translated. Please see English version of document.



```
[:] Set the maximum possible age of a cache lock.  
[:]  
[:] ,  
[:] Experimental  
[:] mod_cache
```

Documentation not yet translated. Please see English version of document.



```
┆ Set the lock path directory.  
┆  
┆ ,  
┆ Experimental  
┆ mod_cache
```

Documentation not yet translated. Please see English version of document.



## CacheMaxExpire

```
CacheMaxExpire seconds  
CacheMaxExpire 86400 ()  
,  
Experimental  
mod_cache
```

CacheMaxExpire

HTTP

,

.

CacheMaxExpire 604800



- [:](#) The minimum time in seconds to cache a document
- [:](#)
- [:](#) , , directory, .htaccess
- [:](#) Experimental
- [:](#) mod\_cache

Documentation not yet translated. Please see English version of document.



```
⋮ Run the cache from the quick handler.  
⋮  
⋮ ,  
⋮ Experimental  
⋮ mod_cache
```

Documentation not yet translated. Please see English version of document.



- [:~](#) Serve stale content in place of 5xx responses.
- [:~](#)
- [:~](#) , , directory, .htaccess
- [:~](#) Experimental
- [:~](#) mod\_cache

Documentation not yet translated. Please see English version of document.



- Attempt to cache responses that the server reports as expired
- 
- , , directory, .htaccess
- Experimental
- mod\_cache

Documentation not yet translated. Please see English version of document.



[\[:\]](#) Attempt to cache requests or responses that have been marked as no-store.

[\[:\]](#)

[\[:\]](#) , , directory, .htaccess

[\[:\]](#) Experimental

[\[:\]](#) mod\_cache

Documentation not yet translated. Please see English version of document.



[:](#) Attempt to cache responses that the server has marked as private

[:](#)

[:](#) , , directory, .htaccess

[:](#) Experimental

[:](#) mod\_cache

Documentation not yet translated. Please see English version of document.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_cache\_disk

- Content cache storage manager keyed to URIs
- Experimental
- cache\_disk\_module
- mod\_cache\_disk.c

. ...

mod\_cache\_disk

mod\_

URI

:  
mod\_cache\_disk mod\_cache



## CacheDirLength

```
CacheDirLength length  
CacheDirLength 2  
,  
Experimental  
mod_cache_disk
```

CacheDirLength .

CacheDirLevels CacheDirLength 20 .

CacheDirLength 4



## CACHEDIRLEVELS

```
CacheDirLevels levels
CacheDirLevels 3
Experimental
mod_cache_disk
```

CacheDirLevels .

CacheDirLevels CacheDirLength 20 .

CacheDirLevels 5



## CACHEMAXFILESIZE

```
:| ()  
:| CacheMaxFileSize bytes  
:| CacheMaxFileSize 1000000  
:| ,  
:| Experimental  
:| mod_cache_disk
```

CacheMaxFileSize .

```
CacheMaxFileSize 64000
```





[:](#) The minimum size (in bytes) of the document to read and be cached before sending the data downstream

[:](#)

[:](#) , , directory, .htaccess

[:](#) Experimental

[:](#) mod\_cache\_disk

Documentation not yet translated. Please see English version of document.



## CacheReadTime

- The minimum time (in milliseconds) that should elapse while reading before data is sent downstream
- 
- , , directory, .htaccess
- Experimental
- mod\_cache\_disk

Documentation not yet translated. Please see English version of document.



## CACHE ROOT

```
┆ root
┆ CacheRoot directory
┆ ,
┆ Experimental
┆ mod_cache_disk
```

```
CacheRoot . mod_c
CacheDirLevels CacheDirLength root
CacheRoot
```

```
CacheRoot c:/cacheroot
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_cache_socache`

<b>Description:</b>	Shared object cache (socache) based storage module for the HTTP caching filter.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>cache_socache_module</code>
<b>Source File:</b>	<code>mod_cache_socache.c</code>

## Summary

`mod_cache_socache` implements a shared object cache (socache) based storage manager for `mod_cache`.

The headers and bodies of cached responses are combined, and stored underneath a single key in the shared object cache. A [number of implementations](#) of shared object caches are available to choose from.

Multiple content negotiated responses can be stored concurrently, however the caching of partial content is not yet supported by this module.

```
# Turn on caching
CacheSocache shmcb
CacheSocacheMaxSize 102400
<Location "/foo">
    CacheEnable socache
</Location>

# Fall back to the disk cache
CacheSocache shmcb
CacheSocacheMaxSize 102400
<Location "/foo">
    CacheEnable socache
    CacheEnable disk
```

```
</Location>
```

**Note:**

[mod\\_cache\\_socache](#) requires the services of [mod\\_cache](#), which must be loaded before [mod\\_cache\\_socache](#).

**Bugfix checklist**

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

**See also**

[mod\\_cache](#)

[mod\\_cache\\_disk](#)

[Caching Guide](#)



<b>Description:</b>	The shared object cache implementation to use
<b>Syntax:</b>	<code>CacheSocache <i>type[:args]</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_cache_socache</code>
<b>Compatibility:</b>	Available in Apache 2.4.5 and later

The `CacheSocache` directive defines the name of the shared object cache implementation to use, followed by optional arguments for that implementation. A [number of implementations](#) of shared object caches are available to choose from.

```
CacheSocache shmcb
```



<b>Description:</b>	The maximum size (in bytes) of an entry to be placed in the cache
<b>Syntax:</b>	CacheSocacheMaxSize <i>bytes</i>
<b>Default:</b>	CacheSocacheMaxSize 102400
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_cache_socache
<b>Compatibility:</b>	Available in Apache 2.4.5 and later

The `CacheSocacheMaxSize` directive sets the maximum size, in bytes, for the combined headers and body of a document to be considered for storage in the cache. The larger the headers that are stored alongside the body, the smaller the body may be.

The `mod_cache_socache` module will only attempt to cache responses that have an explicit content length, or that are small enough to be written in one pass. This is done to allow the `mod_cache_disk` module to have an opportunity to cache responses larger than those cacheable within `mod_cache_socache`.

```
CacheSocacheMaxSize 102400
```



<b>Description:</b>	The maximum time (in seconds) for a document to be placed in the cache
<b>Syntax:</b>	CacheSocacheMaxTime <i>seconds</i>
<b>Default:</b>	CacheSocacheMaxTime 86400
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_cache_socache
<b>Compatibility:</b>	Available in Apache 2.4.5 and later

The `CacheSocacheMaxTime` directive sets the maximum freshness lifetime, in seconds, for a document to be stored in the cache. This value overrides the freshness lifetime defined for the document by the HTTP protocol.

```
CacheSocacheMaxTime 86400
```



<b>Description:</b>	The minimum time (in seconds) for a document to be placed in the cache
<b>Syntax:</b>	CacheSocacheMinTime <i>seconds</i>
<b>Default:</b>	CacheSocacheMinTime 600
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_cache_socache
<b>Compatibility:</b>	Available in Apache 2.4.5 and later

The `CacheSocacheMinTime` directive sets the amount of seconds beyond the freshness lifetime of the response that the response should be cached for in the shared object cache. If a response is only stored for its freshness lifetime, there will be no opportunity to revalidate the response to make it fresh again.

```
CacheSocacheMinTime 600
```



<b>Description:</b>	The minimum size (in bytes) of the document to read and be cached before sending the data downstream
<b>Syntax:</b>	CacheSocacheReadSize <i>bytes</i>
<b>Default:</b>	CacheSocacheReadSize 0
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_cache_socache
<b>Compatibility:</b>	Available in Apache 2.4.5 and later

The `CacheSocacheReadSize` directive sets the minimum amount of data, in bytes, to be read from the backend before the data is sent to the client. The default of zero causes all data read of any size to be passed downstream to the client immediately as it arrives. Setting this to a higher value causes the disk cache to buffer at least this amount before sending the result to the client. This can improve performance when caching content from a slow reverse proxy.

This directive only takes effect when the data is being saved to the cache, as opposed to data being served from the cache.

```
CacheSocacheReadSize 102400
```



<b>Description:</b>	The minimum time (in milliseconds) that should elapse while reading before data is sent downstream
<b>Syntax:</b>	CacheSocacheReadTime <i>milliseconds</i>
<b>Default:</b>	CacheSocacheReadTime 0
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_cache_socache
<b>Compatibility:</b>	Available in Apache 2.4.5 and later

The `CacheSocacheReadTime` directive sets the minimum amount of elapsed time that should pass before making an attempt to send data downstream to the client. During the time period, data will be buffered before sending the result to the client. This can improve performance when caching content from a reverse proxy.

The default of zero disables this option.

This directive only takes effect when the data is being saved to the cache, as opposed to data being served from the cache. It is recommended that this option be used alongside the `CacheSocacheReadSize` directive to ensure that the server does not buffer excessively should data arrive faster than expected.

```
CacheSocacheReadTime 1000
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_cern\_meta

- [:\_ CERN
- [:\_ Extension
- [:\_ cern\_meta\_module
- [:\_ mod\_cern\_meta.c

CERN .  
, Expires:  
CERN .

HTTP

[CERN metafile semantics](#) .



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[mod\\_headers](#)

[mod\\_asis](#)



Module

<a href="#">:</a>	CERN
<a href="#">:</a>	MetaDir <i>directory</i>
<a href="#">:</a>	MetaDir .web
<a href="#">:</a>	, , directory, .htaccess
<b>Override :</b>	Indexes
<a href="#">:</a>	Extension
<a href="#">:</a>	mod_cern_meta

. " :  
:

MetaDir .

:

MetaDir .meta



## MetaFiles

<input type="checkbox"/>	CERN
<input type="checkbox"/>	MetaFiles on off
<input type="checkbox"/>	MetaFiles off
<input type="checkbox"/>	, , directory, .htaccess
<b>Override :</b>	Indexes
<input type="checkbox"/>	Extension
<input type="checkbox"/>	mod_cern_meta

.



<a href="#">:</a>	CERN
<a href="#">:</a>	MetaSuffix <i>suffix</i>
<a href="#">:</a>	MetaSuffix <i>.meta</i>
<a href="#">:</a>	, , directory, .htaccess
<b><a href="#">Override :</a></b>	Indexes
<a href="#">:</a>	Extension
<a href="#">:</a>	mod_cern_meta

. ,  
 DOCUMENT\_ROOT/somedir/index.html  
 DOCUMENT\_ROOT/somedir/.web/index.html.meta  
 MIME .

```

:
MetaSuffix .meta

```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_cgi

- CGI
- Base
- cgi\_module
- mod\_cgi.c

```
mime type    application/x-httpd-cgi    (1.1)
script       CGI ,,
,           ScriptAlias           CGI.
```

```
CGI         DOCUMENT_ROOT .           DocumentRo
```

```
CGI         CGI .
```

```
MPM         mod_cgid .
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[AcceptPathInfo](#)

[Options](#)

[ScriptAlias](#)

[AddHandler](#)

[ID CGI](#)

[CGI](#)



CGI CGI :

### PATH\_INFO

AcceptPathInfo off .  
AcceptPathInfo 404  
NOT FOUND , mod\_cgi (URI  
/more/path/info). AcceptPathInfo m  
AcceptPathInfo On .

### REMOTE\_HOST

HostnameLookups on ( off), DNS

### REMOTE\_IDENT

IdentityCheck on, ident .

### REMOTE\_USER

CGI .



( ) CGI

## CGI

CGI CGI . CGI  
: :

```
%% []  
%% HTTP- CGI--
```

CGI :

```
%%error
```

( ) , :

```
%request  
  HTTP  
( ) POST PUT  
%response  
  CGI  
%stdout  
  CGI  
%stderr  
  CGI
```

( %stdout %stderr ).



## ScriptLog

```
ScriptLog CGI  
ScriptLog file-path  
,  
Base  
mod_cgi, mod_cgid
```

```
ScriptLog CGI . ScriptLog .  
CGI . ServerRoot .
```

```
ScriptLog logs/cgi_log
```

```
, User .  
,
```

```
CGI  
,
```



ScriptLogBuffer

```
PUT POST  
ScriptLogBuffer bytes  
ScriptLogBuffer 1024  
,  
Base  
mod_cgi, mod_cgid
```

. 1024 ,

PUT POST



## ScriptLogLength

```
ScriptLogLength CGI
ScriptLogLength bytes
ScriptLogLength 10385760
,
Base
mod_cgi, mod_cgid
```

```
ScriptLogLength CGI .CGI (
)
CGI .
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_cgid

- CGI
- CGI
- Base
- cgid\_module
- mod\_cgid.c
- MPMs

ScriptSock mod\_cgid mod\_cgi .  
mod\_cgi .

CGI (fork)  
mod\_cgid CGI .  
(unix domain socket) .

MPM mod\_cgi .  
mod\_cgi . cgi



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[mod\\_cgi](#)

[ID CGI](#)



- [:~](#) The length of time to wait for more output from the CGI program
- [:~](#) `CGIDScriptTimeout time[s|ms]`
- [:~](#) value of [Timeout](#) directive when unset
- [:~](#) , , directory, .htaccess
- [:~](#) Base
- [:~](#) mod\_cgid
- [:~](#) CGIDScriptTimeout defaults to zero in releases 2.4 and earlier

The documentation for this directive has not been translated yet. Please have a look at the English version.



## ScriptSock

```
ScriptSock cgi
ScriptSock file-path
ScriptSock logs/cgisock
ScriptSock ,
ScriptSock Base
ScriptSock mod_cgid
```

```
CGI . (root)
.CGI .
```

```
ScriptSock /var/run/cgid.sock
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_charset\_lite

- Experimental
- charset\_lite\_module
- mod\_charset\_lite.c

mod\_charset\_lite .

mod\_charset\_lite

mod\_charset\_lite .

mod\_charset\_lite EBCDIC ASCII

. EBCDIC

ISO-8859-1 .

mod

. ASCII

mod\_charset .



---

mod\_charset\_lite

ARP

CharsetSource

CharsetDefault .

. APR iconv(3) ,

iconv

:

```
iconv -f charsetsourceenc-value -t charsetdefault-value
```

:

•

.

•

(,) .



## CharsetDefault

```
CharsetDefault charset
, , directory, .htaccess
Override : FileInfo
Experimental
mod_charset_lite
```

### CharsetDefault

*charset* APR . iconv

```
<Directory /export/home/trawick/apacheinst/htdocs/convert>
  CharsetSourceEnc UTF-16BE
  CharsetDefault ISO-8859-1
</Directory>
```



## CharsetOptions

```
CharsetOptions option [option] ...
CharsetOptions DebugLevel=0
NoImplicitAdd
, , directory, .htaccess
Override : FileInfo
Experimental
mod_charset_lite
```

CharsetOptions mod\_charset\_lite . Opt

### DebugLevel=*n*

DebugLevel mod\_charset\_lite .  
. DebugLevel=0 .  
. mod\_charset\_lite.c  
.

### ImplicitAdd | NoImplicitAdd

ImplicitAdd mod  
. AddOutputFilter , NoIr  
mod\_charset\_lite .



```

:
: CharsetSourceEnc charset
: , , directory, .htaccess
Override : FileInfo
: Experimental
: mod_charset_lite

```

## CharsetSourceEnc

*charset* APR

. iconv

```

<Directory /export/home/trawick/apacheinst/htdocs/convert>
  CharsetSourceEnc UTF-16BE
  CharsetDefault ISO-8859-1
</Directory>

```

## Solaris 8 iconv



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_data`

<b>Description:</b>	Convert response body into an RFC2397 data URL
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>data_module</code>
<b>Source File:</b>	<code>mod_data.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

This module provides the ability to convert a response into an [RFC2397 data URL](#).

Data URLs can be embedded inline within web pages using something like the [mod\\_include](#) module, to remove the need for clients to make separate connections to fetch what may potentially be many small images. Data URLs may also be included into pages generated by scripting languages such as PHP.

### An example of a data URL

```
data:image/gif;base64,R0lGODdhMAAwAPAAAAAAP///ywAAAAAMAAwAAAC8IyPqcvt3wCcDkiLc7C0qwyGHhSwpjQu5yqmCYsapyuvUUlvONm0ZtfzgFzByTB10QgxOR0TqBQejhRNz0fkVJ+5YiUqrXF5Y5lKh/DeuNcP5yLWGsEbtLi0Spa/TPg7JpJHxyendzWTBfX0cx0nKPjgBzi4diinWGdkF8kjdfnycQZXZeYGejmJlZeG19i2icVqaNVailT6F5iJ90m6mvuTS40K05M0vDk0Q4XUtwwK0zrcd3iq9uisF81M10IcR71Eewwclp7tuNNkM3uNna3F2JQFo97Vriy/Xl4/f1cf5VwzXyym7PHhX4dbgYKAAA7
```

The filter takes no parameters, and can be added to the filter stack using the [SetOutputFilter](#) directive, or any of the directives supported by the [mod\\_filter](#) module.

### Configuring the filter

```
<Location "/data/images">
```

```
SetOutputFilter DATA
</Location>
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Filters](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_dav

- ⌵ Distributed Authoring and Versioning ([WebDAV](#))
- ⌵ Extension
- ⌵ dav\_module
- ⌵ mod\_dav.c

[WebDAV](#) ('Web-based Distributed Authoring and Versioning') class 1 class 2 . WebDAV

( ; ) , ,

(c  
H



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[DavLockDB](#)

[LimitXMLRequestBody](#)

[WebDAV](#)



[mod\\_dav](#) httpd.conf :

```
Dav On
```

```
mod\_dav\_fs DAV (provider) .  
LoadModule .
```

```
, DAV (lock) httpd.conf DavLockDB  
:
```

```
DavLockDB /usr/local/apache2/var/DavLock
```

```
User Group .
```

```
DAV <Location> <Limit> .  
DAV LimitXI  
.  
" LimitRequestBody DAV .
```

```
DavLockDB /usr/local/apache2/var/DavLock
```

```
<Location /foo>  
  Dav On  
  
  AuthType Basic  
  AuthName DAV  
  AuthUserFile user.passwd  
  
  <LimitExcept GET OPTIONS>  
    require user admin  
  </LimitExcept>  
</Location>
```

[mod\\_dav](#) Greg Stein [Apache 1.3 mod\\_dav](#) .



DAV , [mod\\_dav](#)

.

DAV . HTTP Basic Authentic  
. [mod\\_auth\\_digest](#) HTTP Digest Authenticati  
. WebDAV . [SSL](#) B  
Authentication .

[mod\\_dav](#) , [User Group](#) .

, [User Group](#) . . DAV  
( FTP )

.

[mod\\_dav](#) . [LimitXMLRequestBo](#)  
DAV . [DavDepthInfinity](#)

PROPFIND .

. . DAV



---

(PHP, CGI) m  
GET  
URL, URL DAV .

```
Alias /phparea /home/gstein/php_files  
Alias /php-source /home/gstein/php_files  
<Location /php-source>  
    DAV On  
    ForceType text/plain  
</Location>
```

http://example.com/phparea PHP ,  
http://example.com/php-source DAV .



- WebDAV HTTP
- Dav On|Off|*provider-name*
- Dav Off
- directory
- Extension
- mod\_dav

WebDAV HTTP

Dav :

```
<Location /foo>  
  Dav On  
</Location>
```

On mod\_dav fs  
DAV DAV

filesystem.  
.

```
WebDAV .  
.
```





```

: DAV
: DavMinTimeout seconds
: DavMinTimeout 0
: , , directory
: Extension
: mod_dav

```

DAV (lock)

**DavMinTimeout**  
Folders 120 .

() . Microsoft Web  
**DavMinTimeout** (600 )

```

<Location /MSWord>
  DavMinTimeout 600
</Location>

```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_dav\_fs

[: mod\\_dav](#)

[: Extension](#)

[: dav\\_fs\\_module](#)

[: mod\\_dav\\_fs.c](#)

[mod\\_dav](#) . [mod\\_dav](#) .  
(provider) filesystem. [Dav](#) [mod\\_dav](#)

Dav filesystem

filesystem [mod\\_dav](#) On .



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[mod\\_dav](#)



```

: DAV
: DavLockDB file-path
: ,
: Extension
: mod_dav_fs

```

**DavLockDB**

mod\_dav\_fs SDBM

```

DavLockDB var/DavLock

```

User Group

DavLock



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_dav_lock`

<b>Description:</b>	Generic locking module for <code>mod_dav</code>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>dav_lock_module</code>
<b>Source File:</b>	<code>mod_dav_lock.c</code>
<b>Compatibility:</b>	Available in version 2.1 and later

## Summary

This module implements a generic locking API which can be used by any backend provider of `mod_dav`. It *requires* at least the service of `mod_dav`. But without a backend provider which makes use of it, it's useless and should not be loaded into the server. A sample backend module which actually utilizes `mod_dav_lock` is `mod_dav_svn`, the subversion provider module.

Note that `mod_dav_fs` does *not* need this generic locking module, because it uses its own more specialized version.

In order to make `mod_dav_lock` functional, you just have to specify the location of the lock database using the `DavGenericLockDB` directive described below.

## Developer's Note

In order to retrieve the pointer to the locking provider function, you have to use the `ap_lookup_provider` API with the arguments `dav-lock`, `generic`, and `0`.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_dav](#)



<b>Description:</b>	Location of the DAV lock database
<b>Syntax:</b>	<code>DavGenericLockDB</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_dav_lock</code>

Use the `DavGenericLockDB` directive to specify the full path to the lock database, excluding an extension. If the path is not absolute, it will be interpreted relative to `ServerRoot`. The implementation of `mod_dav_lock` uses a SDBM database to track user locks.

### Example

```
DavGenericLockDB var/DavLock
```

The directory containing the lock database file must be writable by the `User` and `Group` under which Apache is running. For security reasons, you should create a directory for this purpose rather than changing the permissions on an existing directory. In the above example, Apache will create files in the `var/` directory under the `ServerRoot` with the base filename `DavLock` and an extension added by the server.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_dbd

<b>Description:</b>	Manages SQL database connections
<b>Status:</b>	Extension
<b>Module Identifier:</b>	dbd_module
<b>Source File:</b>	mod_dbd.c
<b>Compatibility:</b>	Version 2.1 and later

## Summary

[mod\\_dbd](#) manages SQL database connections using [APR](#). It provides database connections on request to modules requiring SQL database functions, and takes care of managing databases with optimal efficiency and scalability for both threaded and non-threaded MPMs. For details, see the [APR](#) website and this overview of the [Apache DBD Framework](#) by its original developer.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Password Formats](#)



This module manages database connections, in a manner optimised for the platform. On non-threaded platforms, it provides a persistent connection in the manner of classic LAMP (Linux, Apache, Mysql, Perl/PHP/Python). On threaded platform, it provides an altogether more scalable and efficient *connection pool*, as described in [this article at ApacheTutor](#). Note that `mod_dbd` supersedes the modules presented in that article.



To connect to your database, you'll need to specify a driver, and connection parameters. These vary from one database engine to another. For example, to connect to mysql, do the following:

```
DBDriver mysql
DBDParams host=localhost,dbname=pony,user=st
```

You can then use this connection in a variety of other modules, including [mod\\_rewrite](#), [mod\\_authn\\_dbd](#), and [mod\\_lua](#). Further usage examples appear in each of those modules' documentation.

See [DBDParams](#) for connection string information for each of the supported database drivers.



mod\_dbd exports five functions for other modules to use. The API is as follows:

```
typedef struct {
    apr_dbd_t *handle;
    apr_dbd_driver_t *driver;
    apr_hash_t *prepared;
} ap_dbd_t;

/* Export functions to access the database

/* acquire a connection that MUST be explicitly closed.
 * Returns NULL on error
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_open(apr_pool_t *pool, const char *driver,
    const char *url);

/* release a connection acquired with ap_dbd_open
 */
AP_DECLARE(void) ap_dbd_close(server_rec *s, ap_dbd_t *dbd);

/* acquire a connection that will have the driver specified in the
 * configuration and MUST NOT be explicitly closed. Returns NULL on
 * error. This is the preferred function for most applications.
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_acquire(server_rec *s);

/* acquire a connection that will have the driver specified in the
 * configuration and MUST NOT be explicitly closed. Returns NULL on
 * error. This is the preferred function for most applications.
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_cacquire(server_rec *s);

/* Prepare a statement for use by a client
 */
AP_DECLARE(void) ap_dbd_prepare(server_rec *s, ap_dbd_t *dbd,
    const char *statement);

/* Also export them as optional functions for modules that don't
 */
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_open, (apr_pool_t*, const char*, const char*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_close, (server_rec*, ap_dbd_t*));
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_acquire, (server_rec*));
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_cacquire, (server_rec*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepare, (server_rec*, ap_dbd_t*, const char*));
```

```
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_close,  
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_a  
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_c  
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepar
```



## SQL Prepared Statements

`mod_dbd` supports SQL prepared statements on behalf of modules that may wish to use them. Each prepared statement must be assigned a name (label), and they are stored in a hash: the prepared field of an `ap_dbd_t`. Hash entries are of type `apr_dbd_prepared_t` and can be used in any of the `apr_dbd` prepared statement SQL query or select commands.

It is up to dbd user modules to use the prepared statements and document what statements can be specified in `httpd.conf`, or to provide their own directives and use `ap_dbd_prepare`.

### **Caveat**

When using prepared statements with a MySQL database, it is preferred to set `reconnect` to 0 in the connection string as to avoid errors that arise from the MySQL client reconnecting without properly resetting the prepared statements. If set to 1, any broken connections will be attempted fixed, but as `mod_dbd` is not informed, the prepared statements will be invalidated.



Any web/database application needs to secure itself against SQL injection attacks. In most cases, Apache DBD is safe, because applications use prepared statements, and untrusted inputs are only ever used as data. Of course, if you use it via third-party modules, you should ascertain what precautions they may require.

However, the *FreeTDS* driver is inherently **unsafe**. The underlying library doesn't support prepared statements, so the driver emulates them, and the untrusted input is merged into the SQL statement.

It can be made safe by *untainting* all inputs: a process inspired by Perl's taint checking. Each input is matched against a regexp, and only the match is used, according to the Perl idiom:

```
$untrusted =~ /([a-z]+)/;  
$trusted = $1;
```

To use this, the untainting regexps must be included in the prepared statements configured. The regexp follows immediately after the % in the prepared statement, and is enclosed in curly brackets {}. For example, if your application expects alphanumeric input, you can use:

```
"SELECT foo FROM bar WHERE input = %s"
```

with other drivers, and suffer nothing worse than a failed query. But with FreeTDS you'd need:

```
"SELECT foo FROM bar WHERE input = %{{{[A-Za-z0-9]+}}}s"
```

Now anything that doesn't match the regexp's \$1 match is discarded, so the statement is safe.

An alternative to this may be the third-party ODBC driver, which offers the security of genuine prepared statements.



<b>Description:</b>	Keepalive time for idle connections
<b>Syntax:</b>	DBDExptime <i>time-in-seconds</i>
<b>Default:</b>	DBDExptime 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

Set the time to keep idle connections alive when the number of connections specified in DBDKeep has been exceeded (threaded platforms only).



<b>Description:</b>	Execute an SQL statement after connecting to a database
<b>Syntax:</b>	DBDInitSQL <i>"SQL statement"</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

Modules, that wish it, can have one or more SQL statements executed when a connection to a database is created. Example usage could be initializing certain values or adding a log entry when a new connection is made to the database.



<b>Description:</b>	Maximum sustained number of connections
<b>Syntax:</b>	DBDKeep <i>number</i>
<b>Default:</b>	DBDKeep 2
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

Set the maximum number of connections per process to be sustained, other than for handling peak demand (threaded platforms only).



<b>Description:</b>	Maximum number of connections
<b>Syntax:</b>	DBDMax <i>number</i>
<b>Default:</b>	DBDMax 10
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

Set the hard maximum number of connections per process (threaded platforms only).



<b>Description:</b>	Minimum number of connections
<b>Syntax:</b>	DBDMin <i>number</i>
<b>Default:</b>	DBDMin 1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

Set the minimum number of connections per process (threaded platforms only).



**Description:** Parameters for database connection

**Syntax:** DBDParams  
*param1=value1 [, param2=value2]*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_dbd

As required by the underlying driver. Typically this will be used to pass whatever cannot be defaulted amongst username, password, database name, hostname and port number for connection.

Connection string parameters for current drivers include:

### **FreeTDS (for MSSQL and SyBase)**

username, password, appname, dbname, host, charset, lang, server

### **MySQL**

host, port, user, pass, dbname, sock, flags, fldsz, group, reconnect

### **Oracle**

user, pass, dbname, server

### **PostgreSQL**

The connection string is passed straight through to PQconnectdb

### **SQLite2**

The connection string is split on a colon, and part1:part2 is used as sqlite\_open(part1, atoi(part2), NULL)

### **SQLite3**

The connection string is passed straight through to sqlite3\_open

### **ODBC**

datasource, user, password, connect, ctimeout, stimeout,  
access, txmode, bufsize



<b>Description:</b>	Whether to use persistent connections
<b>Syntax:</b>	DBDPersist On Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

If set to Off, persistent and pooled connections are disabled. A new database connection is opened when requested by a client, and closed immediately on release. This option is for debugging and low-usage servers.

The default is to enable a pool of persistent connections (or a single LAMP-style persistent connection in the case of a non-threaded server), and should almost always be used in operation.

Prior to version 2.2.2, this directive accepted only the values 0 and 1 instead of Off and On, respectively.



<b>Description:</b>	Define an SQL prepared statement
<b>Syntax:</b>	<code>DBDPrepareSQL "SQL statement" label</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dbd

For modules such as authentication that repeatedly use a single SQL statement, optimum performance is achieved by preparing the statement at startup rather than every time it is used. This directive prepares an SQL statement and assigns it a label.



<b>Description:</b>	Specify an SQL driver
<b>Syntax:</b>	<code>DBDriver <i>name</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_dbd</code>

Selects an `apr_dbd` driver by name. The driver must be installed on your system (on most systems, it will be a shared object or dll). For example, `DBDriver mysql` will select the MySQL driver in `apr_dbd_mysql.so`.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_deflate

```

:
: Extension
: deflate_module
: mod_deflate.c
```

mod\_deflate

DE



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)



## type

AddOutputFilterByType DEFLATE text/html text/plain text/xml

```
<Location />
#
SetOutputFilter DEFLATE

# Netscape 4.x ...
BrowserMatch ^Mozilla/4 gzip-only-text/html

# Netscape 4.06-4.08
BrowserMatch ^Mozilla/4\.0[678] no-gzip

# MSIE Netscape ,
# BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

# : 2.0.48 mod_setenvif
# .
# :
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

#
SetEnvIfNoCase Request_URI \
  \.(?:gif|jpe?g|png)$ no-gzip dont-vary

#
Header append Vary User-Agent env=!dont-vary
</Location>
```



DEFLATE . :

```
SetOutputFilter DEFLATE
```

```
text/html 1 . 1 . html ( )
```

MIME type [AddOutputFilterByType](#) .  
html :

```
<Directory "/your-server-root/manual">  
  AddOutputFilterByType DEFLATE text/html  
</Directory>
```

[BrowserMatch](#)  
no-gzip gzip-only-text/html .  
:

```
BrowserMatch ^Mozilla/4 gzip-only-text/html  
BrowserMatch ^Mozilla/4\.0[678] no-gzip  
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
```

```
User-Agent Netscape Navigator 4.x .  
text/html type . 4.06, 4.07, 4.08 htm  
deflate .
```

[BrowserMatch](#) Microsoft Internet Explorer "Mozilla/4"  
user agent . User  
(\b "" ) .

```
DEFLATE PHP SSI RESOURCE . ,
```

(subrequest)

`SetEnv` force-gzip accept-encoding

`mod_deflate` gzip  
`SetOutputFilter` `AddOutputFilter` INI

```
<Location /dav-area>  
  ProxyPass http://example.com/  
  SetOutputFilter INFLATE  
</Location>
```

example.com gzip

`mod_deflate` gzip  
`SetInputFilter` `AddInputFilter` DEFLATE

```
<Location /dav-area>  
  SetInputFilter DEFLATE  
</Location>
```

Content-Encoding: gzip . gzip

[WebDAV](#)

### Content-Length

, *Content-Length* ! Content-Length



mod\_deflate

Accept-Encoding:

Vary: Accept-Encoding HTTP

, User-Agent

Vary . ,

:

,

User-Agent

DEFLATI

Header append Vary User-Agent

( , HTTP ) ,

Vary \*

Header set Vary \*



```
zlib
DeflateBufferSize value
DeflateBufferSize 8096
,
Extension
mod_deflate
```

DeflateBufferSize zlib .



## DeflateCompressionLevel

```
DeflateCompressionLevel value  
Zlib's default  
,  
Extension  
mod_deflate  
2.0.45
```

```
DeflateCompressionLevel . ,  
. ,  
( ) 1 ( ) 9 .
```



```

:
: DeflateFilterNote [type] notename
: ,
: Extension
: mod_deflate
: type 2.0.4

```

## DeflateFilterNote

```
DeflateFilterNote ratio
```

```
LogFormat "%r" %b (%{ratio}n) "%{User-agent}i" deflate
CustomLog logs/deflate_log deflate
```

```
type . type :
```

### Input

### Output

### Ratio

```
( output/input * 100). type .
```

```
:
```

```
DeflateFilterNote Input instream
DeflateFilterNote Output outstream
DeflateFilterNote Ratio ratio
```

```
LogFormat "%r" %{outstream}n/%{instream}n (%{ratio}n%%)'
deflate
```

CustomLog logs/deflate\_log deflate

- mod\_log\_config



## DeflateInflateLimitRequestBody

- [:](#) Maximum size of inflated request bodies
- [:](#) DeflateInflateLimitRequestBody*value*
- [:](#) None, but LimitRequestBody applies after deflation
- [:](#) , , directory, .htaccess
- [:](#) Extension
- [:](#) mod\_deflate
- [:](#) 2.4.10 and later

The documentation for this directive has not been translated yet.  
Please have a look at the English version.



## DeflateInflateRatioBurst

- [:](#) Maximum number of times the inflation ratio for request bodies can be crossed
- [:](#) DeflateInflateRatioBurst *value*
- [:](#) 3
- [:](#) , , directory, .htaccess
- [:](#) Extension
- [:](#) mod\_deflate
- [:](#) 2.4.10 and later

The documentation for this directive has not been translated yet. Please have a look at the English version.



```
: Maximum inflation ratio for request bodies
: DeflateInflateRatioLimit value
: 200
: , , directory, .htaccess
: Extension
: mod_deflate
: 2.4.10 and later
```

The documentation for this directive has not been translated yet.  
Please have a look at the English version.



## DeflateMemLevel

```
zlib
DeflateMemLevel value
DeflateMemLevel 9
,
Extension
mod_deflate
```

DeflateMemLevel zlib . (1 9



```

: Zlib window size
: DeflateWindowSize value
: DeflateWindowSize 15
: ,
: Extension
: mod_deflate

```

`DeflateWindowSize` zlib window size (1 15 )  
window size .



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_dialup

<b>Description:</b>	Send static content at a bandwidth rate limit, defined by the various old modem standards
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	dialup_module
<b>Source File:</b>	mod_dialup.c

### Summary

It is a module that sends static content at a bandwidth rate limit, defined by the various old modem standards. So, you can browse your site with a 56k V.92 modem, by adding something like this:

```
<Location "/mysite">  
    ModemStandard "V.92"  
</Location>
```

Previously to do bandwidth rate limiting modules would have to block an entire thread, for each client, and insert sleeps to slow the bandwidth down. Using the new suspend feature, a handler can get callback N milliseconds in the future, and it will be invoked by the Event MPM on a different thread, once the timer hits. From there the handler can continue to send data to the client.



<b>Description:</b>	Modem standard to simulate
<b>Syntax:</b>	ModemStandard V.21 V.26bis V.32 V.34 V.92
<b>Context:</b>	directory
<b>Status:</b>	Experimental
<b>Module:</b>	mod_dialup

Specify what modem standard you wish to simulate.

```
<Location "/mysite">  
    ModemStandard "V.26bis"  
</Location>
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_dir

```
┆ " "  
┆ index  
┆ Base  
┆ dir_module  
┆ mod_dir.c
```

index :

- index.html . [DirectoryIndex](#) .  
  [mod\\_dir](#) .
- . [mod\\_autoindex](#) .

index () .

dirname URL http://servername/foo/dirna  
" " .  
http://servername/foo/dirname/ .



## DirectoryCheckHandler

<a href="#">:</a>	Toggle how this module responds when another handler is configured
<a href="#">:</a>	DirectoryCheckHandler On Off
<a href="#">:</a>	DirectoryCheckHandler Off
<a href="#">:</a>	, , directory, .htaccess
<b>Override :</b>	Indexes
<a href="#">:</a>	Base
<a href="#">:</a>	mod_dir
<a href="#">:</a>	Available in 2.4.8 and later. Releases prior to 2.4 implicitly act as if "DirectoryCheckHandler ON" was specified.

The documentation for this directive has not been translated yet. Please have a look at the English version.



## DirectoryIndex

```
DirectoryIndex local-url [local-url]
...
DirectoryIndex index.html
, , directory, .htaccess
Override : Indexes
Base
mod_dir
```

DirectoryIndex / index  
. Local-url (% ) URL. Inc

```
DirectoryIndex index.html
```

http://myserver/docs/  
http://myserver/docs/index.html ,  
.

```
DirectoryIndex index.html index.txt /cgi-bin/index.pl
```

index.html index.txt CGI /cgi-  
bin/index.pl .



## DirectoryIndexRedirect

<code>DirectoryIndexRedirect</code>	Configures an external redirect for directory indexes.
<code>DirectoryIndexRedirect</code>	on   off   permanent   temp   seeother   <i>3xx-code</i>
<code>DirectoryIndexRedirect</code>	off
	, , directory, .htaccess
<b>Override :</b>	Indexes
	Base
	mod_dir
	Available in version 2.3.14 and later

The documentation for this directive has not been translated yet. Please have a look at the English version.



## DirectorySlash

```
DirectorySlash On|Off
DirectorySlash On
, , directory, .htaccess
Override : Indexes
Base
mod_dir
2.0.51
```

```
DirectorySlash mod_dir URL
, mod_dir
```

- URL
- mod\_autoindex .
- DirectoryIndex .
- html URL .

```
# !
<Location /some/path>
  DirectorySlash Off
  SetHandler some-handler
</Location>
```

```
(Options +Indexe
mod_autoindex DirectoryIndex (index.html )
URL .
index.html .
```



```

:~ Define a default URL for requests that don't map to a file
:~
:~ , , directory, .htaccess
:~ Base
:~ mod_dir

```

Documentation not yet translated. Please see English version of document.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_dumpio

<b>Description:</b>	Dumps all I/O to error log as desired.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	dumpio_module
<b>Source File:</b>	mod_dumpio.c

### Summary

mod\_dumpio allows for the logging of all input received by Apache and/or all output sent by Apache to be logged (dumped) to the error.log file.

The data logging is done right after SSL decoding (for input) and right before SSL encoding (for output). As can be expected, this can produce extreme volumes of data, and should only be used when debugging problems.



## Enabling dumpio support

To enable the module, it should be compiled and loaded in to your running Apache configuration. Logging can then be enabled or disabled separately for input and output via the below directives. Additionally, `mod_dumpio` needs to be configured to `LogLevel trace7`:

```
LogLevel dumpio:trace7
```



## DumpIOInput Directive

<b>Description:</b>	Dump all input data to the error log
<b>Syntax:</b>	DumpIOInput On Off
<b>Default:</b>	DumpIOInput Off
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOInput is only available in Apache 2.1.3 and later.

Enable dumping of all input.

```
Example  
DumpIOInput On
```



## DumpIOOutput Directive

<b>Description:</b>	Dump all output data to the error log
<b>Syntax:</b>	DumpIOOutput On Off
<b>Default:</b>	DumpIOOutput Off
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOOutput is only available in Apache 2.1.3 and later.

Enable dumping of all output.

```
Example  
DumpIOOutput On
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_echo



-  echo
-  Experimental
-  echo\_module
-  mod\_echo.c
-  Apache 2.0

.  
, .

echo . telnet



```
ProtocolEcho
: echo
: ProtocolEcho On|Off
: ,
: Experimental
: mod_echo
: ProtocolEcho 2.0
.
```

ProtocolEcho echo .

```
ProtocolEcho On
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_env

- CGI SSI
- Base
- env\_module
- mod\_env.c

CGI SSI



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)



## PassEnv

```
PassEnv env-variable [env-variable]  
...  
, , directory, .htaccess  
Override : FileInfo  
Base  
mod_env
```

CGI

SSI .

```
PassEnv LD_LIBRARY_PATH
```



```
SetEnv env-variable value
, , directory, .htaccess
Override : FileInfo
Base
mod_env
```

CGI SSI .

```
SetEnv SPECIAL_PATH /foo/bin
```



## UNSETENV

```
UnsetEnv env-variable [env-variable]
...
, , directory, .htaccess
Override : FileInfo
Base
mod_env
```

CGI SSI .

```
UnsetEnv LD_LIBRARY_PATH
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_example\_hooks

- API
- Experimental
- example\_hooks\_module
- mod\_example\_hooks.c

```
modules/examples      API
mod_example_hooks.c (callback)
    . !
example . "example-hooks-
handler"      example .
```



## Example

example :

1. `--enable-example-hooks` `configure` .
2. (`" make"`).
- :
- A. `cp modules/examples/mod_example_hooks.c`  
`modules/new_module/mod_myexample.c`
- B. `.`
- C. `modules/new_module/config.m4` .
  1. `APACHE_MODPATH_INIT(new_module)` .
  2. `modules/examples/config.m4` `"example_hooks`  
`APACHE_MODULE` .
  3. `"example_hooks"` `myexample` .
  4. `.` `cor`
  5. `C` , `CFLAGS` ,  
`LD_FLAGS, LIBS` . `modules` `confi`
  6. `APACHE_MODPATH_FINISH` .
- D. `module/new_module/Makefile.in` .  
`include $(top_srcdir)/build/special`
- E. `./buildconf` .
- F. `--enable-myexample`



example httpd.conf :

```
<Location /example-hooks-info>  
SetHandler example-hooks-handler  
</Location>
```

[.htaccess](#) , "test.example"

```
AddHandler example-hooks-handler .example
```



## Example

```
API
Example
, , directory,
.htaccess
Experimental
mod_example_hooks
```

```
Example example
example URL
. "Example directive declared here:
YES/NO"
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_expires

Expires Cache-Control

HTTP

Extension

expires\_module

mod\_expires.c

Expires HTTP Cache-Control HTTP max-age .

HTTP

"" ,

Header

max-age

Cache-Control ([RFC 2616, 1](#)

) .



## ExpiresDefault ExpiresByType :

```
ExpiresDefault "<base> [plus] {<num> <type>}*"
ExpiresByType type/encoding "<base> [plus] {<num> <type>}*"

```

<base> :

- access
- now('access')
- modification

plus .<num> [atoi() ]. <1  
:

- years
- months
- weeks
- days
- hours
- minutes
- seconds

, 1 :

```
ExpiresDefault "access plus 1 month"
ExpiresDefault "access plus 4 weeks"
ExpiresDefault "access plus 30 days"

```

'<num> <type>' :

```
ExpiresByType text/html "access plus 1 month 15 days 2 hours"
ExpiresByType image/gif "modification plus 5 hours 3 minutes"

```

(modification)

Expires



## Expires

<a href="#">:</a>	Expires
<a href="#">:</a>	ExpiresActive On Off
<a href="#">:</a>	, , directory, .htaccess
<a href="#">Override :</a>	Indexes
<a href="#">:</a>	Extension
<a href="#">:</a>	mod_expires

```
(, .htaccess .)
Expires Cache-Control .( .htaccess
) off
ExpiresByType ExpiresDefault ( )
.
Expires Cache-Control .
.
```



## ExpiresByType

```
ExpiresByType MIME-type Expires
ExpiresByType MIME-type <code>seconds
ExpiresByType , , directory, .htaccess
Override : Indexes
ExpiresByType Extension
ExpiresByType mod_expires
```

```
( , text/html) Expires Cache-
Control max-age .
Cache-Control: max-age ,
M , A .
M . URL
A .
( , ), .
```

```
:
#
ExpiresActive On
# GIF
ExpiresByType image/gif A2592000
# HTML ExpiresByType text/html M604800
```

```
ExpiresActive On ExpiresDefa
MIME type
```



## ExpiresDefault

```
: ExpiresDefault <code>seconds  
: , , directory, .htaccess  
Override : Indexes  
: Extension  
: mod_expires
```



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_ext\_filter

```
⋮  
⋮ Extension  
⋮ ext_filter_module  
⋮ mod_ext_filter.c
```

mod\_ext\_filter

```
)  
, :
```

- 
- 
- 

,

API

/

mod\_ext\_filter



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)



## type HTML

```
# mod_ext_filter
# /usr/bin/enscript
# text/c HTML
# type text/html
ExtFilterDefine c-to-html mode=output \
  intype=text/c outtype=text/html \
  cmd="/usr/bin/enscript --color -W html -Ec -o - -"

<Directory "/export/home/trawick/apacheinst/htdocs/c">
# core
SetOutputFilter c-to-html

# .c type text/c mod_mime
#
AddType text/c .c

#
# mod_ext_filter
#
ExtFilterOptions DebugLevel=1
</Directory>
```

## content

Note: gzip .

[mod](#)

```
# mod_ext_filter
ExtFilterDefine gzip mode=output cmd=/bin/gzip

<Location /gzipped>
# gzip core
SetOutputFilter gzip

# "Content-Encoding: gzip"
# mod_header
Header set Content-Encoding gzip
</Location>
```

```

# cat
# mod_ext_filter ; cat
# ;
ExtFilterDefine slowdown mode=output cmd=/bin/cat \
    preservescontentlength

<Location />
    # slowdown    core
    #
    SetOutputFilter slowdown;slowdown;slowdown
</Location>

```

## sed

```

#
# mod_ext_filter
#
ExtFilterDefine fixtext mode=output intype=text/html \
    cmd="/bin/sed s/verdana/arial/g"

<Location />
    # fixtext    core
    SetOutputFilter fixtext
</Location>

```

```

#      (IP 192.168.1.31)
# mod_deflate .
# mod_deflate .
ExtFilterDefine tracebefore \
    cmd="/bin/tracefilter.pl /tmp/tracebefore" \
    EnableEnv=trace_this_client

# mod_deflate .
# ftype ,
# AP_FTYPE_RESOURCE mod_deflate **
# . AP_FTYPE_CONTENT_SET
# mod_deflate .
ExtFilterDefine traceafter \
    cmd="/bin/tracefilter.pl /tmp/traceafter" \
    EnableEnv=trace_this_client ftype=21

<Directory /usr/local/docs>

```

```
SetEnvIf Remote_Addr 192.168.1.31 trace_this_client
SetOutputFilter tracebefore;deflate;traceafter
</Directory>
```

```
:
#!/usr/local/bin/perl -w
use strict;

open(SAVE, ">$ARGV[0]")
  or die "can't open $ARGV[0]: $?";

while (<STDIN>) {
  print SAVE $_;
  print $_;
}

close(SAVE);
```



```

:
: ExtFilterDefine filtername parameters
:
: Extension
: mod_ext_filter

```

```

ExtFilterDefine , .

filtername . SetOutputFilter .
. API .
. ,
:

cmd=cmdline
cmd= .
cmd="/bin/myppgm arg1 arg2").
. .
. CGI DOCUMENT_URI,
DOCUMENT_PATH_INFO, QUERY_STRING_UNESCAPED
.

mode=mode
() mode=output . mode=:
. mode=input 2.1 .

intype=imt
media type(, MIME type) .
. intype= type .

outtype=imt
media type(, MIME type) .
media type . , media type .

PreservesContentLength

```

PreservesContentLength content length .  
content length .

***ftype=filtertype***

AP\_FTYPE\_RESOURCE .  
util\_filter.h AP\_FTYPE\_\* .

***disableenv=env***

***enableenv=env***



## ExtFilterOptions

```

: mod_ext_filter
: ExtFilterOptions option [option] ...
: ExtFilterOptions DebugLevel=0 NoLogStderr
: directory
: Extension
: mod_ext_filter
```

ExtFilterOptions mod\_ext\_filter .

### DebugLevel=*n*

```

DebugLevel mod_ext_filter .
. DebugLevel=0 . ,
. mod_ext_filter.c DBGLV
.
: core LogLevel .
```

### LogStderr | NoLogStderr

```

LogStderr .
NoLogStderr .
```

```
ExtFilterOptions LogStderr DebugLevel=0
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_file\_cache

```
└─ Experimental
└─ file_cache_module
└─ mod_file_cache.c
```

mod file cache

mod\_file\_cache  
mod file cache

: CGI

1.3 mod\_mmap\_static .



mod\_file\_cache MMapFile CacheFile

, AIX .

## MMapFile

mod\_file\_cache MMapFile mmap()

mmap().

rdist mv .  
stat()

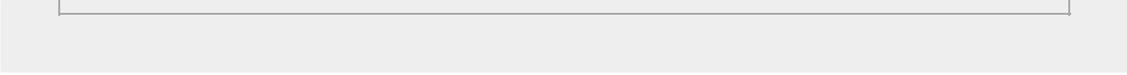
## CacheFile

mod\_file\_cache CacheFile ()

(handle) (file descriptor) .  
API sendfile() ( TransmitFile()).

rdist mv .

```
find /www/htdocs -type f -print \  
| sed -e 's/./mmapfile &/' > /www/conf/mmap.conf
```



## CacheFile

```
CacheFile file-path [file-path] ...  
Experimental  
mod_file_cache
```

CacheFile (open) .  
(close).

file-path . URL-  
stat() inode .  
mod\_rewrite .

```
CacheFile /usr/local/apache/htdocs/index.html
```



## MMapFile

```
┆  
┆ MMapFile file-path [file-path] ...  
┆  
┆ Experimental  
┆ mod_file_cache
```

```
MMapFile ( ) mmap()  
(unmap).  
.  
  
file-path . URL-  
stat() inode .  
mod_rewrite .
```

```
MMapFile /usr/local/apache/htdocs/index.html
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_filter`

<b>Description:</b>	Context-sensitive smart filter configuration module
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>filter_module</code>
<b>Source File:</b>	<code>mod_filter.c</code>
<b>Compatibility:</b>	Version 2.1 and later

## Summary

This module enables smart, context-sensitive configuration of output content filters. For example, apache can be configured to process different content-types through different filters, even when the content-type is not known in advance (e.g. in a proxy).

`mod_filter` works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to `mod_filter`; no change to existing filter modules is required (although it may be possible to simplify them).

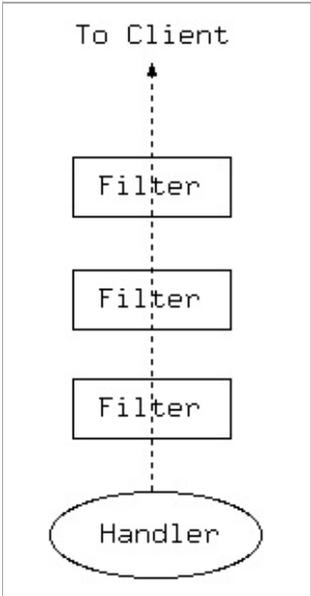


## Smart Filtering

In the traditional filtering model, filters are inserted unconditionally using `AddOutputFilter` and family. Each filter then needs to determine whether to run, and there is little flexibility available for server admins to allow the chain to be configured dynamically.

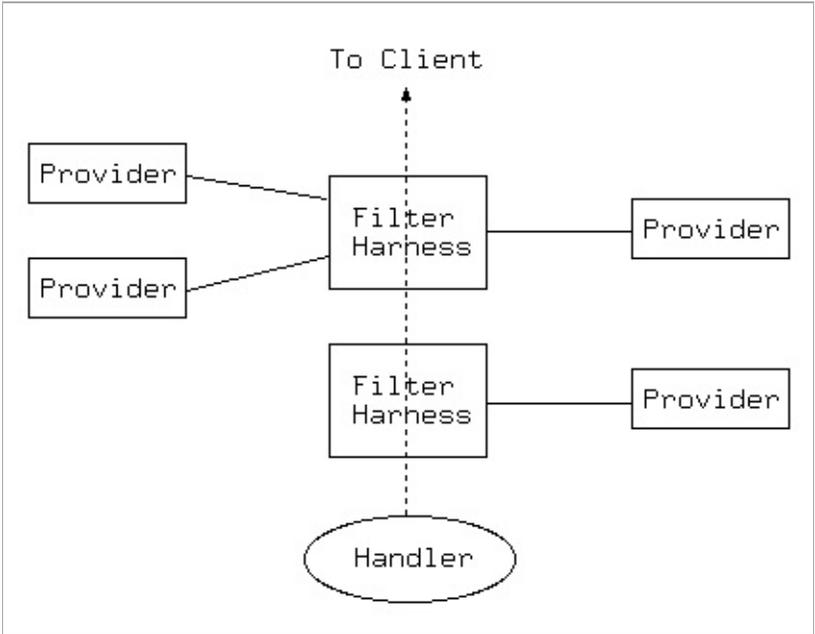
`mod_filter` by contrast gives server administrators a great deal of flexibility in configuring the filter chain. In fact, filters can be inserted based on complex boolean `expressions`. This generalises the limited flexibility offered by `AddOutputFilterByType`.





**Figure 1:** The traditional filter model

In the traditional model, output filters are a simple chain from the content generator (handler) to the client. This works well provided the filter chain can be correctly configured, but presents problems when the filters need to be configured dynamically based on the outcome of the handler.



**Figure 2:** The mod filter model

mod filter works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to mod filter; no change to existing filter modules is required (although it may be possible to simplify them). There can be multiple providers for one filter, but no more than one provider will run for any single request.

A filter chain comprises any number of instances of the filter harness, each of which may have any number of providers. A special case is that of a single provider with unconditional dispatch: this is equivalent to inserting the provider filter directly into the chain.



There are three stages to configuring a filter chain with `mod_filter`. For details of the directives, see below.

### Declare Filters

The `FilterDeclare` directive declares a filter, assigning it a name and filter type. Required only if the filter is not the default type `AP_FTYPE_RESOURCE`.

### Register Providers

The `FilterProvider` directive registers a provider with a filter. The filter may have been declared with `FilterDeclare`; if not, `FilterProvider` will implicitly declare it with the default type `AP_FTYPE_RESOURCE`. The provider must have been registered with `ap_register_output_filter` by some module. The final argument to `FilterProvider` is an expression: the provider will be selected to run for a request if and only if the expression evaluates to true. The expression may evaluate HTTP request or response headers, environment variables, or the Handler used by this request. Unlike earlier versions, `mod_filter` now supports complex expressions involving multiple criteria with AND / OR logic (`&&` / `||`) and brackets. The details of the expression syntax are described in the [ap\\_expr documentation](#).

### Configure the Chain

The above directives build components of a smart filter chain, but do not configure it to run. The `FilterChain` directive builds a filter chain from smart filters declared, offering the flexibility to insert filters at the beginning or end of the chain, remove a filter, or clear the chain.



## Filtering and Response Status

`mod_filter` normally only runs filters on responses with HTTP status 200 (OK). If you want to filter documents with other response statuses, you can set the *filter-errordocs* environment variable, and it will work on all responses regardless of status. To refine this further, you can use expression conditions with `FilterProvider`.



## Configuration

The `FilterProvider` directive has changed from httpd 2.2: the *match* and *dispatch* arguments are replaced with a single but more versatile *expression*. In general, you can convert a match/dispatch pair to the two sides of an expression, using something like:

```
"dispatch = 'match'"
```

The Request headers, Response headers and Environment variables are now interpreted from syntax `%{req:foo}`, `%{resp:foo}` and `%{env:foo}` respectively. The variables `%{HANDLER}` and `%{CONTENT_TYPE}` are also supported.

Note that the match no longer support substring matches. They can be replaced by regular expression matches.



## Server side Includes (SSI)

A simple case of replacing `AddOutputFilterByType`

```
FilterDeclare SSI
FilterProvider SSI INCLUDES "%{CONTENT_T
FilterChain SSI
```

## Server side Includes (SSI)

The same as the above but dispatching on handler (classic SSI behaviour; .shtml files get processed).

```
FilterProvider SSI INCLUDES "%{HANDLER}
FilterChain SSI
```

## Emulating mod\_gzip with mod\_deflate

Insert INFLATE filter only if "gzip" is NOT in the Accept-Encoding header. This filter runs with ftype CONTENT\_SET.

```
FilterDeclare gzip CONTENT_SET
FilterProvider gzip inflate "%{req:Accep
FilterChain gzip
```

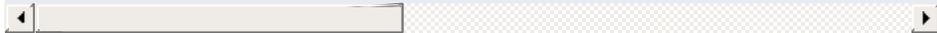
## Image Downsampling

Suppose we want to downsample all web images, and have filters for GIF, JPEG and PNG.

```
FilterProvider unpack jpeg_unpack "%{CON
FilterProvider unpack gif_unpack "%{CONT
FilterProvider unpack png_unpack "%{CONT
```

```
FilterProvider downsample downsample_fil  
FilterProtocol downsample "change=yes"
```

```
FilterProvider repack jpeg_pack "%{CONTE  
FilterProvider repack gif_pack "%{CONTEN  
FilterProvider repack png_pack "%{CONTEN  
<Location "/image-filter">  
    FilterChain unpack downsample repack  
</Location>
```



Historically, each filter is responsible for ensuring that whatever changes it makes are correctly represented in the HTTP response headers, and that it does not run when it would make an illegal change. This imposes a burden on filter authors to re-implement some common functionality in every filter:

- Many filters will change the content, invalidating existing content tags, checksums, hashes, and lengths.
- Filters that require an entire, unbroken response in input need to ensure they don't get byteranges from a backend.
- Filters that transform output in a filter need to ensure they don't violate a `Cache-Control: no-transform` header from the backend.
- Filters may make responses uncacheable.

`mod_filter` aims to offer generic handling of these details of filter implementation, reducing the complexity required of content filter modules. This is work-in-progress; the `FilterProtocol` implements some of this functionality for back-compatibility with Apache 2.0 modules. For `httpd 2.1` and later, the `ap_register_output_filter_protocol` and `ap_filter_protocol` API enables filter modules to declare their own behaviour.

At the same time, `mod_filter` should not interfere with a filter that wants to handle all aspects of the protocol. By default (i.e. in the absence of any `FilterProtocol` directives), `mod_filter` will leave the headers untouched.

At the time of writing, this feature is largely untested, as modules in common use are designed to work with 2.0. Modules using it should test it carefully.



## AddOutputFilterByType Directive

<b>Description:</b>	assigns an output filter to a particular media-type
<b>Syntax:</b>	AddOutputFilterByType <i>filter</i> [ <i>;filter...</i> ] <i>media-type</i> [ <i>media-type</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_filter
<b>Compatibility:</b>	Had severe limitations before being moved to <a href="#">mod_filter</a> in version 2.3.7

This directive activates a particular output [filter](#) for a request depending on the response [media-type](#).

The following example uses the DEFLATE filter, which is provided by [mod\\_deflate](#). It will compress all output (either static or dynamic) which is labeled as text/html or text/plain before it is sent to the client.

```
AddOutputFilterByType DEFLATE text/html text
```

If you want the content to be processed by more than one filter, their names have to be separated by semicolons. It's also possible to use one [AddOutputFilterByType](#) directive for each of these filters.

The configuration below causes all script output labeled as text/html to be processed at first by the INCLUDES filter and then by the DEFLATE filter.

```
<Location "/cgi-bin/">
```

```
Options Includes  
AddOutputFilterByType INCLUDES;DEFLATE 1  
</Location>
```

## See also

- [AddOutputFilter](#)
- [SetOutputFilter](#)
- [filters](#)



<b>Description:</b>	Configure the filter chain
<b>Syntax:</b>	<code>FilterChain [+=-@!]filter-name ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_filter

This configures an actual filter chain, from declared filters. `FilterChain` takes any number of arguments, each optionally preceded with a single-character control that determines what to do:

- +*filter-name***  
Add *filter-name* to the end of the filter chain
- @*filter-name***  
Insert *filter-name* at the start of the filter chain
- filter-name***  
Remove *filter-name* from the filter chain
- =*filter-name***  
Empty the filter chain and insert *filter-name*
- !**  
Empty the filter chain
- filter-name***  
Equivalent to +*filter-name*



<b>Description:</b>	Declare a smart filter
<b>Syntax:</b>	<code>FilterDeclare <i>filter-name</i> [<i>type</i>]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_filter

This directive declares an output filter together with a header or environment variable that will determine runtime configuration. The first argument is a *filter-name* for use in [FilterProvider](#), [FilterChain](#) and [FilterProtocol](#) directives.

The final (optional) argument is the type of filter, and takes values of `ap_filter_type` - namely RESOURCE (the default), CONTENT\_SET, PROTOCOL, TRANSCODE, CONNECTION or NETWORK.



<b>Description:</b>	Deal with correct HTTP protocol handling
<b>Syntax:</b>	<code>FilterProtocol <i>filter-name</i> [<i>provider-name</i>] <i>proto-flags</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_filter

This directs [mod\\_filter](#) to deal with ensuring the filter doesn't run when it shouldn't, and that the HTTP response headers are correctly set taking into account the effects of the filter.

There are two forms of this directive. With three arguments, it applies specifically to a *filter-name* and a *provider-name* for that filter. With two arguments it applies to a *filter-name* whenever the filter runs *any* provider.

Flags specified with this directive are merged with the flags that underlying providers may have registered with [mod\\_filter](#). For example, a filter may internally specify the equivalent of `change=yes`, but a particular configuration of the module can override with `change=no`.

*proto-flags* is one or more of

#### **change=yes | no**

Specifies whether the filter changes the content, including possibly the content length. The "no" argument is supported in 2.4.7 and later.

#### **change=1:1**

The filter changes the content, but will not change the content length

**byteranges=no**

The filter cannot work on byteranges and requires complete input

**proxy=no**

The filter should not run in a proxy context

**proxy=transform**

The filter transforms the response in a manner incompatible with the HTTP Cache-Control: no-transform header.

**cache=no**

The filter renders the output uncacheable (eg by introducing randomised content changes)



<b>Description:</b>	Register a content filter
<b>Syntax:</b>	<code>FilterProvider <i>filter-name</i> <i>provider-name</i> <i>expression</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_filter

This directive registers a *provider* for the smart filter. The provider will be called if and only if the *expression* declared evaluates to true when the harness is first called.

*provider-name* must have been registered by loading a module that registers the name with `ap_register_output_filter`.

*expression* is an [ap\\_expr](#).

## See also

- [Expressions in Apache HTTP Server](#), for a complete reference and examples.
- [mod\\_include](#)



<b>Description:</b>	Get debug/diagnostic information from <a href="#">mod_filter</a>
<b>Syntax:</b>	FilterTrace <i>filter-name level</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_filter

This directive generates debug information from [mod\\_filter](#). It is designed to help test and debug providers (filter modules), although it may also help with [mod\\_filter](#) itself.

The debug output depends on the *level* set:

#### **0 (default)**

No debug information is generated.

#### **1**

[mod\\_filter](#) will record buckets and brigades passing through the filter to the error log, before the provider has processed them. This is similar to the information generated by [mod\\_diagnostics](#).

#### **2 (not yet implemented)**

Will dump the full data passing through to a tempfile before the provider. **For single-user debug only**; this will not support concurrent hits.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_headers

- ⌵ HTTP
- ⌵ Extension
- ⌵ headers\_module
- ⌵ mod\_headers.c
- ⌵ RequestHeader  
2.0

HTTP

· , ·



mod\_headers

.

,

```
RequestHeader append MirrorID "mirror 12"  
RequestHeader unset MirrorID
```

```
MirrorID . MirrorID "mirror 12"
```

.



(early) (late)

mod\_headers

(late)

(early) / .

early

, ,

<Directory> <Location>



## 1. "TS" .

```
Header echo ^TS
```

## 2. MyHeader

```
Header add MyHeader "%D %t"
```

```
MyHeader: D=3775428 t=991424704447256
```

## 3. Joe

```
Header add MyHeader "Hello Joe. It took %D microseconds \\  
for Apache to serve this request."
```

```
MyHeader: Hello Joe. It took D=3775428 microseconds for  
Apache to serve this request.
```

## 4. "MyRequestHeader" MyHeader mod\_setenvif

```
SetEnvIf MyRequestHeader value HAVE_MyRequestHeader  
Header add MyHeader "%D %t mytext"  
env=HAVE_MyRequestHeader
```

```
HTTP MyRequestHeader: value ,
```

```
MyHeader: D=3775428 t=991424704447256 mytext
```



## Header

```
: HTTP
: Header [condition]
: set|append|add|unset|echo header
: [value] [early|env=[!]variable]
: , , directory, .htaccess
Override : FileInfo
: Extension
: mod_headers
```

```
HTTP ,.
.
condition , onsuccess always .
. onsuccess 2xx , always ( 2x:
, .
.
set . value .
append .
. HTTP .
add . () .
append .
unset .
.
echo . header .
```

```

header .
add, unset . echo header . set,
add, append, set value . value
. value ,
.

```

%%	
%t	epoch (1970 1 1) t= .
%D	...
%{FOOBAR}e	FOOBAR .
%{FOOBAR}s	<u>mod_ssl</u> , <u>SSL</u> FOOBAR .

```

%s 2.1 . SSLOptions +Sto
    %e . SSLOptions
, %e %s .

```

```

Header .
. env=... ( env=!...
Header .
Header .
Header .

```



## RequestHeader

```
[:] HTTP
[:] RequestHeader set|append|add|unset
header [value] [early|env=[!]variable]
[:] , , directory, .htaccess
Override : FileInfo
[:] Extension
[:] mod_headers
```

```
HTTP , .
. . .
set
.
append
. HTTP .
add
. () .
append .
unset
.
.
.
append, set value . value
unset value . value ,
. Header .
RequestHeader
. env=... ( env=
) RequestHeader .
```

fixup

Requesti

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_heartbeat`

<b>Description:</b>	Sends messages with server status to frontend proxy
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>heartbeat_module</code>
<b>Source File:</b>	<code>mod_heartbeat</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

### Summary

`mod_heartbeat` sends multicast messages to a `mod_heartmonitor` listener that advertises the servers current connection count. Usually, `mod_heartmonitor` will be running on a proxy server with `mod_lbmethod_heartbeat` loaded, which allows `ProxyPass` to use the "heartbeat" `lbmethod` inside of `ProxyPass`.

`mod_heartbeat` itself is loaded on the origin server(s) that serve requests through the proxy server(s).

To use `mod_heartbeat`, `mod_status` and `mod_watchdog` must be either a static modules or, if a dynamic module, must be loaded before `mod_heartbeat`.



## Generating the Feedback Output

Every 1 second, this module generates a single multicast UDP packet, containing the number of busy and idle workers. The packet is a simple ASCII format, similar to GET query parameters in HTTP.

### An Example Packet

```
v=1&ready=75&busy=0
```

Consumers should handle new variables besides busy and ready, separated by '&', being added in the future.



<b>Description:</b>	Multicast address for heartbeat packets
<b>Syntax:</b>	HeartbeatAddress <i>addr:port</i>
<b>Default:</b>	disabled
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_heartbeat

The `HeartbeatAddress` directive specifies the multicast address to which `mod_heartbeat` will send status information. This address will usually correspond to a configured `HeartbeatListen` on a frontend proxy system.

```
HeartbeatAddress 239.0.0.1:27999
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_heartmonitor`

<b>Description:</b>	Centralized monitor for <code>mod_heartbeat</code> origin servers
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>heartmonitor_module</code>
<b>Source File:</b>	<code>mod_heartmonitor.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

### Summary

`mod_heartmonitor` listens for server status messages generated by `mod_heartbeat` enabled origin servers and makes their status available to `mod_lbmethod_heartbeat`. This allows `ProxyPass` to use the "heartbeat" `lbmethod` inside of `ProxyPass`.

This module uses the services of `mod_slotmem_shm` when available instead of flat-file storage. No configuration is required to use `mod_slotmem_shm`.

To use `mod_heartmonitor`, `mod_status` and `mod_watchdog` must be either a static modules or, if a dynamic module, it must be loaded before `mod_heartmonitor`.



<b>Description:</b>	multicast address to listen for incoming heartbeat requests
<b>Syntax:</b>	<code>HeartbeatListenaddr:port</code>
<b>Default:</b>	disabled
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_heartmonitor

The `HeartbeatListen` directive specifies the multicast address on which the server will listen for status information from `mod_heartbeat`-enabled servers. This address will usually correspond to a configured `HeartbeatAddress` on an origin server.

```
HeartbeatListen 239.0.0.1:27999
```

This module is inactive until this directive is used.



<b>Description:</b>	Specifies the maximum number of servers that will be sending heartbeat requests to this server
<b>Syntax:</b>	HeartbeatMaxServers <i>number-of-servers</i>
<b>Default:</b>	HeartbeatMaxServers 10
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_heartmonitor

The `HeartbeatMaxServers` directive specifies the maximum number of servers that will be sending requests to this monitor server. It is used to control the size of the shared memory allocated to store the heartbeat info when `mod_slotmem_shm` is in use.



## HeartbeatStorage Directive

<b>Description:</b>	Path to store heartbeat data
<b>Syntax:</b>	HeartbeatStorage <i>file-path</i>
<b>Default:</b>	HeartbeatStorage logs/hb.dat
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_heartmonitor

The **HeartbeatStorage** directive specifies the path to store heartbeat data. This flat-file is used only when **mod\_slotmem\_shm** is not loaded.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_http2

<b>Description:</b>	Support for the HTTP/2 transport layer
<b>Status:</b>	Extension
<b>Module Identifier:</b>	http2_module
<b>Source File:</b>	mod_http2.c
<b>Compatibility:</b>	Available in version 2.4.17 and later

## Summary

This module provides HTTP/2 ([RFC 7540](#)) support for the Apache HTTP Server.

This module relies on [libnghttp2](#) to provide the core http/2 engine.

You must enable HTTP/2 via [Protocols](#) in order to use the functionality described in this document. The HTTP/2 protocol [does not require](#) the use of encryption so two schemes are available: h2 (HTTP/2 over TLS) and h2c (HTTP/2 over TCP).

Two useful configuration schemes are:

### HTTP/2 in a VirtualHost context (TLS only)

```
Protocols h2 http/1.1
```

Allows HTTP/2 negotiation (h2) via TLS ALPN in a secure [<VirtualHost>](#). HTTP/2 preamble checking (Direct mode, see [H2Direct](#)) is disabled by default for h2.

### HTTP/2 in a Server context (TLS and cleartext)

```
Protocols h2 h2c http/1.1
```

Allows HTTP/2 negotiation (h2) via TLS ALPN for secure [<VirtualHost>](#). Allows HTTP/2 cleartext negotiation (h2c) upgrading from an initial HTTP/1.1 connection or via HTTP/2 preamble checking (Direct mode, see [H2Direct](#)).

Refer to the official [HTTP/2 FAQ](#) for any doubt about the protocol.



## HTTP/2 Dimensioning

Enabling HTTP/2 on your Apache Server has impact on the resource consumption and if you have a busy site, you may need to consider carefully the implications.

The first noticeable thing after enabling HTTP/2 is that your server processes will start additional threads. The reason for this is that HTTP/2 gives all requests that it receives to its own *Worker* threads for processing, collects the results and streams them out to the client.

In the current implementation, these workers use a separate thread pool from the MPM workers that you might be familiar with. This is just how things are right now and not intended to be like this forever. (It might be forever for the 2.4.x release line, though.) So, HTTP/2 workers, or shorter H2Workers, will not show up in `mod_status`. They are also not counted against directives such as `ThreadsPerChild`. However they take `ThreadsPerChild` as default if you have not configured something else via `H2MinWorkers` and `H2MaxWorkers`.

Another thing to watch out for is memory consumption. Since HTTP/2 keeps more state on the server to manage all the open request, priorities for and dependencies between them, it will always need more memory than HTTP/1.1 processing. There are three directives which steer the memory footprint of a HTTP/2 connection: `H2MaxSessionStreams`, `H2WindowSize` and `H2StreamMaxMemSize`.

`H2MaxSessionStreams` limits the number of parallel requests that a client can make on a HTTP/2 connection. It depends on your site how many you should allow. The default is 100 which is plenty and unless you run into memory problems, I would keep it

this way. Most requests that browsers send are GETs without a body, so they use up only a little bit of memory until the actual processing starts.

[H2WindowSize](#) controls how much the client is allowed to send as body of a request, before it waits for the server to encourage more. Or, the other way around, it is the amount of request body data the server needs to be able to buffer. This is per request.

And last, but not least, [H2StreamMaxMemSize](#) controls how much response data shall be buffered. The request sits in a H2Worker thread and is producing data, the HTTP/2 connection tries to send this to the client. If the client does not read fast enough, the connection will buffer this amount of data and then suspend the H2Worker.

## Multiple Hosts and Misdirected Requests

Many sites use the same TLS certificate for multiple virtual hosts. The certificate either has a wildcard name, such as '\*.example.org' or carries several alternate names. Browsers using HTTP/2 will recognize that and reuse an already opened connection for such hosts.

While this is great for performance, it comes at a price: such vhosts need more care in their configuration. The problem is that you will have multiple requests for multiple hosts on the same TLS connection. And that makes renegotiation impossible, in face the HTTP/2 standard forbids it.

So, if you have several virtual hosts using the same certificate and want to use HTTP/2 for them, you need to make sure that all vhosts have exactly the same SSL configuration. You need the same protocol, ciphers and settings for client verification.

If you mix things, Apache httpd will detect it and return a special

response code, 421 Misdirected Request, to the client.

## Environment Variables

This module can be configured to provide HTTP/2 related information as additional environment variables to the SSI and CGI namespace, as well as in custom log configurations (see % {VAR\_NAME}e).

<b>Variable Name:</b>	<b>Value Type:</b>	<b>Description:</b>
HTTP2	flag	HTTP/2 is being used.
H2PUSH	flag	HTTP/2 Server Push is enabled for this connection and also supported by the client.
H2_PUSH	flag	alternate name for H2PUSH
H2_PUSHED	string	empty or PUSHED for a request being pushed by the server.
H2_PUSHED_ON	number	HTTP/2 stream number that triggered the push of this request.
H2_STREAM_ID	number	HTTP/2 stream number of this request.
H2_STREAM_TAG	string	HTTP/2 process unique stream identifier, consisting of connection id and stream id separated by - .



## H2CopyFiles Directive

<b>Description:</b>	Determine file handling in responses
<b>Syntax:</b>	H2CopyFiles on off
<b>Default:</b>	H2CopyFiles off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.24 and later.

This directive influences how file content is handled in responses. When `off`, which is the default, file handles are passed from the request processing down to the main connection, using the usual Apache `setaside` handling for managing the lifetime of the file.

When set to `on`, file content is copied while the request is still being processed and the buffered data is passed on to the main connection. This is better if a third party module is injecting files with different lifetimes into the response.

An example for such a module is `mod_wsgi` that may place Python file handles into the response. Those files get close down when Python thinks processing has finished. That may be well before `mod_http2` is done with them.



<b>Description:</b>	H2 Direct Protocol Switch
<b>Syntax:</b>	H2Direct on off
<b>Default:</b>	H2Direct on for h2c, off for h2 protocol
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2

This directive toggles the usage of the HTTP/2 Direct Mode. This should be used inside a [<VirtualHost>](#) section to enable direct HTTP/2 communication for that virtual host.

Direct communication means that if the first bytes received by the server on a connection match the HTTP/2 preamble, the HTTP/2 protocol is switched to immediately without further negotiation. This mode is defined in RFC 7540 for the cleartext (h2c) case. Its use on TLS connections not mandated by the standard.

When a server/vhost does not have h2 or h2c enabled via [Protocols](#), the connection is never inspected for a HTTP/2 preamble. **H2Direct** does not matter then. This is important for connections that use protocols where an initial read might hang indefinitely, such as NNTP.

For clients that have out-of-band knowledge about a server supporting h2c, direct HTTP/2 saves the client from having to perform an HTTP/1.1 upgrade, resulting in better performance and avoiding the Upgrade restrictions on request bodies.

This makes direct h2c attractive for server to server communication as well, when the connection can be trusted or is secured by other means.

## Example

H2Direct on



## HTTP/2 Early Hints

<b>Description:</b>	Determine sending of 103 status codes
<b>Syntax:</b>	H2EarlyHints on off
<b>Default:</b>	H2EarlyHints off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.24 and later.

This setting controls if HTTP status 103 interim responses are forwarded to the client or not. By default, this is currently not the case since a range of clients still have trouble with unexpected interim responses.

When set to on, PUSH resources announced with H2PushResource will trigger an interim 103 response before the final response. The 103 response will carry Link headers that advise the preload of such resources.



**Description:** Maximum number of active streams per HTTP/2 session.

**Syntax:** `H2MaxSessionStreams n`

**Default:** `H2MaxSessionStreams 100`

**Context:** server config, virtual host

**Status:** Extension

**Module:** `mod_http2`

This directive sets the maximum number of active streams per HTTP/2 session (e.g. connection) that the server allows. A stream is active if it is not `idle` or `closed` according to RFC 7540.

### Example

```
H2MaxSessionStreams 20
```



**Description:** Maximum number of seconds h2 workers remain idle until shut down.

**Syntax:** H2MaxWorkerIdleSeconds *n*

**Default:** H2MaxWorkerIdleSeconds 600

**Context:** server config

**Status:** Extension

**Module:** mod\_http2

This directive sets the maximum number of seconds a h2 worker may idle until it shuts itself down. This only happens while the number of h2 workers exceeds [H2MinWorkers](#).

### Example

```
H2MaxWorkerIdleSeconds 20
```



**Description:** Maximum number of worker threads to use per child process.

**Syntax:** `H2MaxWorkers n`

**Context:** server config

**Status:** Extension

**Module:** `mod_http2`

This directive sets the maximum number of worker threads to spawn per child process for HTTP/2 processing. If this directive is not used, `mod_http2` will chose a value suitable for the mpm module loaded.

### Example

```
H2MaxWorkers 20
```



<b>Description:</b>	Minimal number of worker threads to use per child process.
<b>Syntax:</b>	H2MinWorkers <i>n</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2

This directive sets the minimum number of worker threads to spawn per child process for HTTP/2 processing. If this directive is not used, [mod\\_http2](#) will chose a value suitable for the mpm module loaded.

### Example

```
H2MinWorkers 10
```



<b>Description:</b>	Require HTTP/2 connections to be "modern TLS" only
<b>Syntax:</b>	H2ModernTLSOnly on off
<b>Default:</b>	H2ModernTLSOnly on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.18 and later.

This directive toggles the security checks on HTTP/2 connections in TLS mode (https:). This can be used server wide or for specific [<VirtualHost>s](#).

The security checks require that the TSL protocol is at least TLSv1.2 and that none of the ciphers listed in RFC 7540, Appendix A is used. These checks will be extended once new security requirements come into place.

The name stems from the [Security/Server Side TLS](#) definitions at mozilla where "modern compatibility" is defined. Mozilla Firefox and other browsers require modern compatibility for HTTP/2 connections. As everything in OpSec, this is a moving target and can be expected to evolve in the future.

One purpose of having these checks in [mod\\_http2](#) is to enforce this security level for all connections, not only those from browsers. The other purpose is to prevent the negotiation of HTTP/2 as a protocol should the requirements not be met.

Ultimately, the security of the TLS connection is determined by the server configuration directives for [mod\\_ssl](#).

## Example

H2ModernTLSonly off



<b>Description:</b>	H2 Server Push Switch
<b>Syntax:</b>	H2Push on off
<b>Default:</b>	H2Push on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.18 and later.

This directive toggles the usage of the HTTP/2 server push protocol feature.

The HTTP/2 protocol allows the server to push other resources to a client when it asked for a particular one. This is helpful if those resources are connected in some way and the client can be expected to ask for it anyway. The pushing then saves the time it takes the client to ask for the resources itself. On the other hand, pushing resources the client never needs or already has is a waste of bandwidth.

Server pushes are detected by inspecting the Link headers of responses (see <https://tools.ietf.org/html/rfc5988> for the specification). When a link thus specified has the `rel=preload` attribute, it is treated as a resource to be pushed.

Link headers in responses are either set by the application or can be configured via [mod\\_headers](#) as:

### **mod\_headers example**

```
<Location /index.html>
  Header add Link "</css/site.css>;rel=preload"
  Header add Link "</images/logo.jpg>;rel=preload"
</Location>
```

As the example shows, there can be several link headers added to a response, resulting in several pushes being triggered. There are no checks in the module to avoid pushing the same resource twice or more to one client. Use with care.

HTTP/2 server pushes are enabled by default. This directive allows it to be switch off on all resources of this server/virtual host.

### Example

```
H2Push off
```

Last but not least, pushes happen only when the client signals its willingness to accept those. Most browsers do, some, like Safari 9, do not. Also, pushes also only happen for resources from the same *authority* as the original response is for.



<b>Description:</b>	H2 Server Push Diary Size
<b>Syntax:</b>	H2PushDiarySize <i>n</i>
<b>Default:</b>	H2PushDiarySize 256
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.19 and later.

This directive toggles the maximum number of HTTP/2 server pushes that are remembered per HTTP/2 connection. This can be used inside the `<VirtualHost>` section to influence the number for all connections to that virtual host.

The push diary records a digest (currently using a 64 bit number) of pushed resources (their URL) to avoid duplicate pushes on the same connection. These value are not persisted, so clients opening a new connection will experience known pushes again. There is ongoing work to enable a client to disclose a digest of the resources it already has, so the diary maybe initialized by the client on each connection setup.

If the maximum size is reached, newer entries replace the oldest ones. A diary entry uses 8 bytes, letting a default diary with 256 entries consume around 2 KB of memory.

A size of 0 will effectively disable the push diary.



<b>Description:</b>	H2 Server Push Priority
<b>Syntax:</b>	H2PushPriority <i>mime-type</i> [after before interleaved] [weight]
<b>Default:</b>	H2PushPriority * After 16
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.18 and later. For having an effect, a nghttp2 library version 1.5.0 or newer is necessary.

This directive defines the priority handling of pushed responses based on the content-type of the response. This is usually defined per server config, but may also appear in a virtual host.

HTTP/2 server pushes are always related to a client request. Each such request/response pairs, or *streams* have a dependency and a weight, together defining the *priority* of a stream.

When a stream *depends* on another, say X depends on Y, then Y gets all bandwidth before X gets any. Note that this does not mean that Y will block X. If Y has no data to send, all bandwidth allocated to Y can be used by X.

When a stream has more than one dependant, say X1 and X2 both depend on Y, the *weight* determines the bandwidth allocation. If X1 and X2 have the same weight, they both get half of the available bandwidth. If the weight of X1 is twice as large as that for X2, X1 gets twice the bandwidth of X2.

Ultimately, every stream depends on the *root* stream which gets all the bandwidth available, but never sends anything. So all its bandwidth is distributed by weight among its children. Which either

have data to send or distribute the bandwidth to their own children. And so on. If none of the children have data to send, that bandwidth get distributed somewhere else according to the same rules.

The purpose of this priority system is to always make use of available bandwidth while allowing precedence and weight to be given to specific streams. Since, normally, all streams are initiated by the client, it is also the one that sets these priorities.

Only when such a stream results in a PUSH, gets the server to decide what the *initial* priority of such a pushed stream is. In the examples below, X is the client stream. It depends on Y and the server decides to PUSH streams P1 and P2 onto X.

The default priority rule is:

### **Default Priority Rule**

```
H2PushPriority * After 16
```

which reads as 'Send a pushed stream of any content-type depending on the client stream with weight 16'. And so P1 and P2 will be send after X and, as they have equal weight, share bandwidth equally among themselves.

### **Interleaved Priority Rule**

```
H2PushPriority text/css Interleaved 256
```

which reads as 'Send any CSS resource on the same dependency and weight as the client stream'. If P1 has content-type 'text/css', it will depend on Y (as does X) and its effective weight will be calculated as  $P1ew = Xw * (P1w / 256)$ . With P1w being 256, this will make the effective weight the same as the weight of X. If both X and P1 have data to send, bandwidth will be allocated

to both equally.

With Pw specified as 512, a pushed, interleaved stream would get double the weight of X. With 128 only half as much. Note that effective weights are always capped at 256.

### Before Priority Rule

```
H2PushPriority application/json Before
```

This says that any pushed stream of content type 'application/json' should be send out *before* X. This makes P1 dependent on Y and X dependent on P1. So, X will be stalled as long as P1 has data to send. The effective weight is inherited from the client stream. Specifying a weight is not allowed.

Be aware that the effect of priority specifications is limited by the available server resources. If a server does not have workers available for pushed streams, the data for the stream may only ever arrive when other streams have been finished.

Last, but not least, there are some specifics of the syntax to be used in this directive:

1. '\*' is the only special content-type that matches all others. 'image/\*' will not work.
2. The default dependency is 'After'.
3. There are also default weights: for 'After' it is 16, 'interleaved' is 256.

### Shorter Priority Rules

```
H2PushPriority application/json 32          # an After rule
H2PushPriority image/jpeg before          # weight inherited
H2PushPriority text/css    interleaved    # weight 256 default
```



<b>Description:</b>	Declares resources for early pushing to the client
<b>Syntax:</b>	H2PushResource [add] path [critical]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.24 and later.

When added to a directory/location HTTP/2 PUSHes will be attempted for all paths added via this directive. This directive can be used several times for the same location.

This directive pushes resources much earlier than adding Link headers via [mod\\_headers](#). [mod\\_http2](#) announces these resources in a 103 Early Hints interim response to the client. That means that clients not supporting PUSH will still get early preload hints.

In contrast to setting Link response headers via [mod\\_headers](#), this directive will only take effect on HTTP/2 connections.

By adding `critical` to such a resource, the server will give processing it more preference and send its data, once available, before the data from the main request.



<b>Description:</b>	Serialize Request/Response Processing Switch
<b>Syntax:</b>	H2SerializeHeaders on off
<b>Default:</b>	H2SerializeHeaders off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2

This directive toggles if HTTP/2 requests shall be serialized in HTTP/1.1 format for processing by httpd core or if received binary data shall be passed into the request\_recs directly.

Serialization will lower performance, but gives more backward compatibility in case custom filters/hooks need it.

### Example

```
H2SerializeHeaders on
```



**Description:** Maximum amount of output data buffered per stream.

**Syntax:** `H2StreamMaxMemSize` *bytes*

**Default:** `H2StreamMaxMemSize` 65536

**Context:** server config, virtual host

**Status:** Extension

**Module:** `mod_http2`

This directive sets the maximum number of outgoing data bytes buffered in memory for an active streams. This memory is not allocated per stream as such. Allocations are counted against this limit when they are about to be done. Stream processing freezes when the limit has been reached and will only continue when buffered data has been sent out to the client.

### Example

```
H2StreamMaxMemSize 128000
```



<b>Description:</b>	
<b>Syntax:</b>	H2TLSCoolDownSecs <i>seconds</i>
<b>Default:</b>	H2TLSCoolDownSecs 1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.18 and later.

This directive sets the number of seconds of idle time on a TLS connection before the TLS write size falls back to small (~1300 bytes) length. This can be used server wide or for specific [<VirtualHost>s](#).

See [H2TLSCoolDownSecs](#) for a description of TLS warmup. [H2TLSCoolDownSecs](#) reflects the fact that connections may deteriorate over time (and TCP flow adjusts) for idle connections as well. It is beneficial to overall performance to fall back to the pre-warmup phase after a number of seconds that no data has been sent.

In deployments where connections can be considered reliable, this timer can be disabled by setting it to 0.

The following example sets the seconds to zero, effectively disabling any cool down. Warmed up TLS connections stay on maximum record size.

### Example

```
H2TLSCoolDownSecs 0
```



<b>Description:</b>	
<b>Syntax:</b>	H2TLSSwarmUpSize <i>amount</i>
<b>Default:</b>	H2TLSSwarmUpSize 1048576
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2
<b>Compatibility:</b>	Available in version 2.4.18 and later.

This directive sets the number of bytes to be sent in small TLS records (~1300 bytes) until doing maximum sized writes (16k) on https: HTTP/2 connections. This can be used server wide or for specific `<VirtualHost>`s.

Measurements by [google performance labs](#) show that best performance on TLS connections is reached, if initial record sizes stay below the MTU level, to allow a complete record to fit into an IP packet.

While TCP adjust its flow-control and window sizes, longer TLS records can get stuck in queues or get lost and need retransmission. This is of course true for all packets. TLS however needs the whole record in order to decrypt it. Any missing bytes at the end will stall usage of the received ones.

After a sufficient number of bytes have been send successfully, the TCP state of the connection is stable and maximum TLS record sizes (16 KB) can be used for optimal performance.

In deployments where servers are reached locally or over reliable connections only, the value might be decreased with 0 disabling any warmup phase altogether.

The following example sets the size to zero, effectively disabling

any warmup phase.

### Example

```
H2TLSSwarmUpSize 0
```



## H2Upgrade Directive

<b>Description:</b>	H2 Upgrade Protocol Switch
<b>Syntax:</b>	H2Upgrade on off
<b>Default:</b>	H2Upgrade on for h2c, off for h2 protocol
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2

This directive toggles the usage of the HTTP/1.1 Upgrade method for switching to HTTP/2. This should be used inside a [<VirtualHost>](#) section to enable Upgrades to HTTP/2 for that virtual host.

This method of switching protocols is defined in HTTP/1.1 and uses the "Upgrade" header (thus the name) to announce willingness to use another protocol. This may happen on any request of a HTTP/1.1 connection.

This method of protocol switching is enabled by default on cleartext (potential h2c) connections and disabled on TLS (potential h2), as mandated by RFC 7540.

Please be aware that Upgrades are only accepted for requests that carry no body. POSTs and PUTs with content will never trigger an upgrade to HTTP/2. See [H2Direct](#) for an alternative to Upgrade.

This mode only has an effect when h2 or h2c is enabled via the [Protocols](#).

### Example

```
H2Upgrade on
```



<b>Description:</b>	Size of Stream Window for upstream data.
<b>Syntax:</b>	H2WindowSize <i>bytes</i>
<b>Default:</b>	H2WindowSize 65535
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_http2

This directive sets the size of the window that is used for flow control from client to server and limits the amount of data the server has to buffer. The client will stop sending on a stream once the limit has been reached until the server announces more available space (as it has processed some of the data).

This limit affects only request bodies, not its meta data such as headers. Also, it has no effect on response bodies as the window size for those are managed by the clients.

### Example

```
H2WindowSize 128000
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_ident

[RFC 1413 ident](#)

[Extension](#)

[ident\\_module](#)

[mod\\_ident.c](#)

[2.1](#)

[RFC 1413](#) .



### Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[mod\\_log\\_config](#)



## Identity Check

- [RFC 1413](#)
- [IdentityCheck On|Off](#)
- [IdentityCheck Off](#)
- [, , directory](#)
- [Extension](#)
- [mod\\_ident](#)
- [2.1 core](#)

[RFC 1413](#) identd  
%...1 .



[IdentityCheckTimeout](#)



## IdentityCheckTimeout

```
:_ ident
:_ IdentityCheckTimeout seconds
:_ IdentityCheckTimeout 30
:_ , , directory
:_ Extension
:_ mod_ident
```

ident .

[RFC](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_imagemap

- ▢ (imagemap)
- ▢ Base
- ▢ imagemap\_module
- ▢ mod\_imagemap.c

```
imagemap CGI .map . ( AddHandler  
SetHandler ) imap-file .  
.map .
```

```
AddHandler imap-file map
```

```
AddType application/x-httpd-imagemap map
```

```
" MIME type"
```



- 
- Referer: URL .
  - base <base> .
  - imagemap.conf .
  - (point) .
  - .



```
directive value [x,y ...]
directive value "Menu text" [x,y ...]
directive value x,y ... "Menu text"
```

directive base, default, poly, circle, rect, point .

value URL URL .

. '#' .

6 . ,

### base

<base href="value"> . URL URL

URL . base .htaccess

ImapBase . ImapBase base

http://server\_name/.

base\_uri base .URL .

### default

poly, circle, rect point

. ImapDefault 204 No Conte

nocontent. .

### poly

. .

### circle

. .

### rect

. .

## point

.  
point

point  
default

value .

## URL

URL URL . URL

'.' ,

base base .

, base mailto:

## map

URL . [ImapMenu](#) none .

## menu

map .

## referer

( ) URL .

Referer:

http://servername/.

## nocontent

204 No Content .

## error

500 Server Error .

base

, default .

0,0 200,200

x y .

0,0

**"Menu Text"**

value

```
<a href="http://foo.com/">Menu text</a>
```

```
<a href="http://foo.com/">http://foo.com</a>
```

&quot; .



```
#'formatted' 'semiformatted' .
# html . <hr>
base referer
poly map " ." 0,0 0,10 10,10 10,0
rect .. 0,0 77,27 " "
circle http://www.inetnebr.com/lincoln/feedback/ 195,0 305,27
rect another_file " " 306,0 419,27
point http://www.zyzyva.com/ 100,100
point http://www.tripod.com/ 200,200
rect mailto:nate@tripod.com 100,150 200,0 "?"
```



---

## HTML

```
<a href="/maps/imagemap1.map">  
    
</a>
```

## XHTML

```
<a href="/maps/imagemap1.map">  
    
</a>
```



```

: base
: ImapBase map|referer|URL
: ImapBase http://servername/
: , , directory, .htaccess
Override : Indexes
: Base
: mod_imagemap

```

```

ImapBase base .
. , base http://servername/.

```

- [UseCanonicalName](#)



## ImapDefault

```
ImapDefault
error|nocontent|map|referer|URL
ImapDefault nocontent
, , directory, .htaccess
Override : Indexes
Base
mod_imagemap
```

```
ImapDefault default .
default . , default
No Content nocontent.
```



<a href="#">:</a>	
<a href="#">:</a>	ImapMenu none   formatted   semiformatted   unformatte
<a href="#">:</a>	, , directory, .htaccess
<a href="#">Override :</a>	Indexes
<a href="#">:</a>	Base
<a href="#">:</a>	mod_imagemap

## ImapMenu

### none

ImapMenu none, default .

### formatted

formatted .  
 , .

### semiformatted

semiformatted . HTML .  
 , formatted .

### unformatted

, .  
 . , .  
 .



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_include

<b>Description:</b>	Server-parsed html documents (Server Side Includes)
<b>Status:</b>	Base
<b>Module Identifier:</b>	include_module
<b>Source File:</b>	mod_include.c

## Summary

This module provides a filter which will process files before they are sent to the client. The processing is controlled by specially formatted SGML comments, referred to as *elements*. These elements allow conditional text, the inclusion of other files or programs, as well as the setting and printing of environment variables.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Options](#)

[AcceptPathInfo](#)

[Filters](#)

[SSI Tutorial](#)



Server Side Includes are implemented by the [INCLUDES filter](#). If documents containing server-side include directives are given the extension `.shtml`, the following directives will make Apache parse them and assign the resulting document the mime type of `text/html`:

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

The following directive must be given for the directories containing the `shtml` files (typically in a [<Directory>](#) section, but this directive is also valid in `.htaccess` files if [AllowOverride Options](#) is set):

```
Options +Includes
```

For backwards compatibility, the server - parsed [handler](#) also activates the `INCLUDES` filter. As well, Apache will activate the `INCLUDES` filter for any document with mime type `text/x-server-parsed-html` or `text/x-server-parsed-html3` (and the resulting output will have the mime type `text/html`).

For more information, see our [Tutorial on Server Side Includes](#).



---

Files processed for server-side includes no longer accept requests with PATH\_INFO (trailing pathname information) by default. You can use the [AcceptPathInfo](#) directive to configure the server to accept requests with PATH\_INFO.



The document is parsed as an HTML document, with special commands embedded as SGML comments. A command has the syntax:

```
<!--#element attribute=value attribute=value ... -->
```

The value will often be enclosed in double quotes, but single quotes (') and backticks (`) are also possible. Many commands only allow a single attribute-value pair. Note that the comment terminator (- ->) should be preceded by whitespace to ensure that it isn't considered part of an SSI token. Note that the leading <! - - # is *one* token and may not contain any whitespaces.

The allowed elements are listed in the following table:

Element	Description
<a href="#">comment</a>	SSI comment
<a href="#">config</a>	configure output formats
<a href="#">echo</a>	print variables
<a href="#">exec</a>	execute external programs
<a href="#">fsize</a>	print size of a file
<a href="#">flastmod</a>	print last modification time of a file
<a href="#">include</a>	include a file
<a href="#">printenv</a>	print all available variables
<a href="#">set</a>	set a value of a variable

SSI elements may be defined by modules other than [mod\\_include](#). In fact, the [exec](#) element is provided by [mod\\_cgi](#), and will only be available if this module is loaded.

## The comment Element

This command doesn't output anything. Its only use is to add comments within a file. These comments are not printed.

This syntax is available in version 2.4.21 and later.

```
<!--#comment Blah Blah Blah -->
```

## The config Element

This command controls various aspects of the parsing. The valid attributes are:

### **echormsg** (*Apache 2.1 and later*)

The value is a message that is sent back to the client if the [echo](#) element attempts to echo an undefined variable. This overrides any [SSIUndefinedEcho](#) directives.

```
<!--#config echormsg="[Value Undefined]" -->
```

### **errormsg**

The value is a message that is sent back to the client if an error occurs while parsing the document. This overrides any [SSIErrorMsg](#) directives.

```
<!--#config errormsg="[Oops, something broke.]" -->
```

### **sizefmt**

The value sets the format to be used when displaying the size of a file. Valid values are `bytes` for a count in bytes, or `abbrev` for a count in Kb or Mb as appropriate, for example a size of 1024 bytes will be printed as "1K".

```
<!--#config sizefmt="abbrev" -->
```

## **timefmt**

The value is a string to be used by the `strftime(3)` library routine when printing dates.

```
<!--#config timefmt=""%R, %B %d, %Y"" -->
```

## **The echo Element**

This command prints one of the [include variables](#) defined below. If the variable is unset, the result is determined by the [SSIUndefinedEcho](#) directive. Any dates printed are subject to the currently configured `timefmt`.

Attributes:

### **var**

The value is the name of the variable to print.

### **decoding**

Specifies whether Apache should strip an encoding from the variable before processing the variable further. The default is none, where no decoding will be done. If set to `url`, then URL decoding (also known as %-encoding; this is appropriate for use within URLs in links, etc.) will be performed. If set to `urlencoded`, application/x-www-form-urlencoded compatible encoding (found in query strings) will be stripped. If set to `base64`, base64 will be decoded, and if set to `entity`, HTML entity encoding will be stripped. Decoding is done prior to any further encoding on the variable. Multiple encodings can be stripped by specifying more than one comma separated encoding. The decoding setting will remain in effect until the next decoding attribute is encountered, or the element ends.

The `decoding` attribute must *precede* the corresponding `var` attribute to be effective.

## encoding

Specifies how Apache should encode special characters contained in the variable before outputting them. If set to `none`, no encoding will be done. If set to `url`, then URL encoding (also known as %-encoding; this is appropriate for use within URLs in links, etc.) will be performed. If set to `urlencode`, application/x-www-form-urlencoded compatible encoding will be performed instead, and should be used with query strings. If set to `base64`, base64 encoding will be performed. At the start of an `echo` element, the default is set to `entity`, resulting in entity encoding (which is appropriate in the context of a block-level HTML element, e.g. a paragraph of text). This can be changed by adding an `encoding` attribute, which will remain in effect until the next `encoding` attribute is encountered or the element ends, whichever comes first.

The `encoding` attribute must *precede* the corresponding `var` attribute to be effective.

In order to avoid cross-site scripting issues, you should *always* encode user supplied data.

### Example

```
<!--#echo encoding="entity" var="QUERY_STRING" -->
```

## The exec Element

The `exec` command executes a given shell command or CGI script. It requires `mod_cgi` to be present in the server. If `Options IncludesNOEXEC` is set, this command is completely disabled. The valid attributes are:

## cgi

The value specifies a (%-encoded) URL-path to the CGI script. If the path does not begin with a slash (/), then it is taken to be relative to the current document. The document referenced by this path is invoked as a CGI script, even if the server would not normally recognize it as such. However, the directory containing the script must be enabled for CGI scripts (with [ScriptAlias](#) or [Options ExecCGI](#)).

The CGI script is given the PATH\_INFO and query string (QUERY\_STRING) of the original request from the client; these *cannot* be specified in the URL path. The include variables will be available to the script in addition to the standard [CGI](#) environment.

### Example

```
<!--#exec cgi="/cgi-bin/example.cgi" -->
```

If the script returns a Location: header instead of output, then this will be translated into an HTML anchor.

The [include virtual](#) element should be used in preference to `exec cgi`. In particular, if you need to pass additional arguments to a CGI program, using the query string, this cannot be done with `exec cgi`, but can be done with `include virtual`, as shown here:

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value" -->
```

## cmd

The server will execute the given string using `/bin/sh`. The [include variables](#) are available to the command, in addition to the usual set of CGI variables.

The use of `#include virtual` is almost always preferred to using either `#exec cgi` or `#exec cmd`. The former (`#include virtual`) uses the standard Apache sub-request mechanism to include files or scripts. It is much better tested and maintained.

In addition, on some platforms, like Win32, and on unix when using `suexec`, you cannot pass arguments to a command in an `exec` directive, or otherwise include spaces in the command. Thus, while the following will work under a non-`suexec` configuration on unix, it will not produce the desired result under Win32, or when running `suexec`:

```
<!--#exec cmd="perl /path/to/perlscript arg1 arg2" -->
```

## The `fsize` Element

This command prints the size of the specified file, subject to the `sizefmt` format specification. Attributes:

### **file**

The value is a path relative to the directory containing the current document being parsed.

```
This file is <!--#fsize file="mod_include.html" --> bytes.
```

The value of `file` cannot start with a slash (`/`), nor can it contain `../` so as to refer to a file above the current directory or outside of the document root. Attempting to so will result in the error message: The given path was above the root path.

### **virtual**

The value is a (%-encoded) URL-path. If it does not begin with a slash (`/`) then it is taken to be relative to the current

document. Note, that this does *not* print the size of any CGI output, but the size of the CGI script itself.

```
This file is <!--#fsize virtual="/docs/mod/mod_include.html" --> bytes.
```

Note that in many cases these two are exactly the same thing. However, the `file` attribute doesn't respect URL-space aliases.

## The `lastmod` Element

This command prints the last modification date of the specified file, subject to the `timefmt` format specification. The attributes are the same as for the `fsize` command.

## The `include` Element

This command inserts the text of another document or file into the parsed file. Any included file is subject to the usual access control. If the directory containing the parsed file has `Options IncludesNOEXEC` set, then only documents with a text `MIME-type` (text/plain, text/html etc.) will be included. Otherwise CGI scripts are invoked as normal using the complete URL given in the command, including any query string.

An attribute defines the location of the document, and may appear more than once in an include element; an inclusion is done for each attribute given to the include command in turn. The valid attributes are:

### **file**

The value is a path relative to the directory containing the current document being parsed. It cannot contain `../`, nor can it be an absolute path. Therefore, you cannot include files that are outside of the document root, or above the current

document in the directory structure. The `virtual` attribute should always be used in preference to this one.

## **virtual**

The value is a (%-encoded) URL-path. The URL cannot contain a scheme or hostname, only a path and an optional query string. If it does not begin with a slash (/) then it is taken to be relative to the current document.

A URL is constructed from the attribute, and the output the server would return if the URL were accessed by the client is included in the parsed output. Thus included files can be nested.

If the specified URL is a CGI program, the program will be executed and its output inserted in place of the directive in the parsed file. You may include a query string in a CGI url:

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value"
-->
```

`include virtual` should be used in preference to `exec cgi` to include the output of CGI programs into an HTML document.

If the [KeptBodySize](#) directive is correctly configured and valid for this included file, attempts to POST requests to the enclosing HTML document will be passed through to subrequests as POST requests as well. Without the directive, all subrequests are processed as GET requests.

## **onerror**

The value is a (%-encoded) URL-path which is shown should a previous attempt to include a file or virtual attribute failed. To be effective, this attribute must be specified after the file or virtual attributes being covered. If the attempt to include the

onerror path fails, or if onerror is not specified, the default error message will be included.

```
# Simple example
<!--#include virtual="/not-exist.html"
onerror="/error.html" -->
```

```
# Dedicated onerror paths
<!--#include virtual="/path-a.html" onerror="/error-
a.html" virtual="/path-b.html" onerror="/error-b.html" -->
```

## The printenv Element

This prints out a plain text listing of all existing variables and their values. Special characters are entity encoded (see the [echo](#) element for details) before being output. There are no attributes.

### Example

```
<pre> <!--#printenv --> </pre>
```

## The set Element

This sets the value of a variable. Attributes:

### **var**

The name of the variable to set.

### **value**

The value to give a variable.

### **decoding**

Specifies whether Apache should strip an encoding from the variable before processing the variable further. The default is none, where no decoding will be done. If set to `url`, `urlencoded`, `base64` or `entity`, URL decoding, application/x-www-form-urlencoded decoding, base64 decoding or HTML entity decoding will be performed

respectively. More than one decoding can be specified by separating with commas. The decoding setting will remain in effect until the next decoding attribute is encountered, or the element ends. The decoding attribute must *precede* the corresponding var attribute to be effective.

## **encoding**

Specifies how Apache should encode special characters contained in the variable before setting them. The default is none, where no encoding will be done. If set to url, urlencoding, base64 or entity, URL encoding, application/x-www-form-urlencoded encoding, base64 encoding or HTML entity encoding will be performed respectively. More than one encoding can be specified by separating with commas. The encoding setting will remain in effect until the next encoding attribute is encountered, or the element ends. The encoding attribute must *precede* the corresponding var attribute to be effective. Encodings are applied after all decodings have been stripped.

### **Example**

```
<!--#set var="category" value="help" -->
```



In addition to the variables in the standard CGI environment, these are available for the `echo` command, for `if` and `elif`, and to any program invoked by the document.

#### **DATE\_GMT**

The current date in Greenwich Mean Time.

#### **DATE\_LOCAL**

The current date in the local time zone.

#### **DOCUMENT\_ARGS**

This variable contains the query string of the active SSI document, or the empty string if a query string is not included. For subrequests invoked through the `include` SSI directive, `QUERY_STRING` will represent the query string of the subrequest and `DOCUMENT_ARGS` will represent the query string of the SSI document. (Available in Apache HTTP Server 2.4.19 and later.)

#### **DOCUMENT\_NAME**

The filename (excluding directories) of the document requested by the user.

#### **DOCUMENT\_URI**

The (%-decoded) URL path of the document requested by the user. Note that in the case of nested include files, this is *not* the URL for the current document. Note also that if the URL is modified internally (e.g. by an [alias](#) or [directoryindex](#)), the modified URL is shown.

#### **LAST\_MODIFIED**

The last modification date of the document requested by the user.

#### **QUERY\_STRING\_UNESCAPED**

If a query string is present in the request for the active SSI document, this variable contains the (%-decoded) query

string, which is *escaped* for shell usage (special characters like & etc. are preceded by backslashes). It is not set if a query string is not present. Use `DOCUMENT_ARGS` if shell escaping is not desired.



## VARIABLE SUBSTITUTION

Variable substitution is done within quoted strings in most cases where they may reasonably occur as an argument to an SSI directive. This includes the `config`, `exec`, `flastmod`, `fsize`, `include`, `echo`, and `set` directives. If [SSILegacyExprParser](#) is set to on, substitution also occurs in the arguments to conditional operators. You can insert a literal dollar sign into the string using backslash quoting:

```
<!--#set var="cur" value="\$test" -->
```

If a variable reference needs to be substituted in the middle of a character sequence that might otherwise be considered a valid identifier in its own right, it can be disambiguated by enclosing the reference in braces, a *la* shell substitution:

```
<!--#set var="Zed" value="{REMOTE_HOST}_{REQUEST_METHOD}" -->
```

This will result in the Zed variable being set to "X\_Y" if REMOTE\_HOST is "X" and REQUEST\_METHOD is "Y".



The basic flow control elements are:

```
<!--#if expr="test_condition" -->  
<!--#elif expr="test_condition" -->  
<!--#else -->  
<!--#endif -->
```

The `if` element works like an if statement in a programming language. The test condition is evaluated and if the result is true, then the text until the next `elif`, `else` or `endif` element is included in the output stream.

The `elif` or `else` statements are used to put text into the output stream if the original *test\_condition* was false. These elements are optional.

The `endif` element ends the `if` element and is required.

*test\_condition* is a boolean expression which follows the [ap\\_expr](#) syntax. The syntax can be changed to be compatible with Apache HTTPD 2.2.x using [SSILegacyExprParser](#).

The SSI variables set with the `var` element are exported into the request environment and can be accessed with the `reqenv` function. As a short-cut, the function name `v` is also available inside [mod\\_include](#).

The below example will print "from local net" if client IP address belongs to the 10.0.0.0/8 subnet.

```
<!--#if expr='-R "10.0.0.0/8"' -->  
    from local net  
<!--#else -->  
    from somewhere else  
<!--#endif -->
```

The below example will print "foo is bar" if the variable foo is set to the value "bar".

```
<!--#if expr='v("foo") = "bar"' -->  
    foo is bar  
<!--#endif -->
```

### Reference Documentation

See also: [Expressions in Apache HTTP Server](#), for a complete reference and examples. The *restricted* functions are not available inside [mod\\_include](#)



This section describes the syntax of the `#if` expr element if [SSILegacyExprParser](#) is set to on.

***string***

true if *string* is not empty

***-A string***

true if the URL represented by the string is accessible by configuration, false otherwise. This is useful where content on a page is to be hidden from users who are not authorized to view the URL, such as a link to that URL. Note that the URL is only tested for whether access would be granted, not whether the URL exists.

**Example**

```
<!--#if expr="-A /private" -->
  Click <a href="/private">here</a> to access private
  information.
<!--#endif -->
```

***string1 = string2***

***string1 == string2***

***string1 != string2***

Compare *string1* with *string2*. If *string2* has the form */string2/* then it is treated as a regular expression. Regular expressions are implemented by the [PCRE](#) engine and have the same syntax as those in [perl 5](#). Note that `==` is just an alias for `=` and behaves exactly the same way.

If you are matching positive (`=` or `==`), you can capture grouped parts of the regular expression. The captured parts are stored in the special variables `$1 .. $9`. The whole string matched by the regular expression is stored in the special variable `$0`

## Example

```
<!--#if expr="$QUERY_STRING = /^sid=([a-zA-Z0-9]+)/" -->  
  <!--#set var="session" value="$1" -->  
<!--#endif -->
```

***string1 < string2***

***string1 <= string2***

***string1 > string2***

***string1 >= string2***

Compare *string1* with *string2*. Note, that strings are compared *literally* (using `strcmp(3)`). Therefore the string "100" is less than "20".

***( test\_condition )***

true if *test\_condition* is true

***! test\_condition***

true if *test\_condition* is false

***test\_condition1 && test\_condition2***

true if both *test\_condition1* and *test\_condition2* are true

***test\_condition1 || test\_condition2***

true if either *test\_condition1* or *test\_condition2* is true

"=" and "!=" bind more tightly than "&&" and "||". "!" binds most tightly. Thus, the following are equivalent:

```
<!--#if expr="$a = test1 && $b = test2" -->  
<!--#if expr="($a = test1) && ($b = test2)" -->
```

The boolean operators `&&` and `||` share the same priority. So if you want to bind such an operator more tightly, you should use parentheses.

Anything that's not recognized as a variable or an operator is treated as a string. Strings can also be quoted: `'string'`.

Unquoted strings can't contain whitespace (blanks and tabs) because it is used to separate tokens such as variables. If multiple strings are found in a row, they are concatenated using blanks. So,

```
string1 string2 results in string1 string2
```

and

```
'string1 string2' results in string1 string2.
```

## Optimization of Boolean Expressions

If the expressions become more complex and slow down processing significantly, you can try to optimize them according to the evaluation rules:

- Expressions are evaluated from left to right
- Binary boolean operators (&& and | |) are short circuited wherever possible. In conclusion with the rule above that means, `mod_include` evaluates at first the left expression. If the left result is sufficient to determine the end result, processing stops here. Otherwise it evaluates the right side and computes the end result from both left and right results.
- Short circuit evaluation is turned off as long as there are regular expressions to deal with. These must be evaluated to fill in the backreference variables (\$1 .. \$9).

If you want to look how a particular expression is handled, you can recompile `mod_include` using the `-DDEBUG_INCLUDE` compiler option. This inserts for every parsed expression tokenizer information, the parse tree and how it is evaluated into the output sent to the client.

## Escaping slashes in regex strings

All slashes which are not intended to act as delimiters in your regex must be escaped. This is regardless of their meaning to the regex engine.



<b>Description:</b>	String that ends an include element
<b>Syntax:</b>	SSIEndTag <i>tag</i>
<b>Default:</b>	SSIEndTag "-->"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_include

This directive changes the string that [mod\\_include](#) looks for to mark the end of an include element.

```
SSIEndTag "%>"
```

## See also

- [SSIStartTag](#)



<b>Description:</b>	Error message displayed when there is an SSI error
<b>Syntax:</b>	SSIErrMsg <i>message</i>
<b>Default:</b>	SSIErrMsg "[an error occurred while processing this directive]"
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Base
<b>Module:</b>	mod_include

The `SSIErrMsg` directive changes the error message displayed when `mod_include` encounters an error. For production servers you may consider changing the default error message to "`<!-- Error -->`" so that the message is not presented to the user.

This directive has the same effect as the `<!--#config errmsg=message -->` element.

```
SSIErrMsg "<!-- Error -->"
```



## SSIETag Directive

<b>Description:</b>	Controls whether ETags are generated by the server.
<b>Syntax:</b>	SSIETag on off
<b>Default:</b>	SSIETag off
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.2.15 and later.

Under normal circumstances, a file filtered by `mod_include` may contain elements that are either dynamically generated, or that may have changed independently of the original file. As a result, by default the server is asked not to generate an ETag header for the response by adding `no-etag` to the request notes.

The `SSIETag` directive suppresses this behaviour, and allows the server to generate an ETag header. This can be used to enable caching of the output. Note that a backend server or dynamic content generator may generate an ETag of its own, ignoring `no-etag`, and this ETag will be passed by `mod_include` regardless of the value of this setting. `SSIETag` can take on the following values:

### **off**

`no-etag` will be added to the request notes, and the server is asked not to generate an ETag. Where a server ignores the value of `no-etag` and generates an ETag anyway, the ETag will be respected.

### **on**

Existing ETags will be respected, and ETags generated by the server will be passed on in the response.



<b>Description:</b>	Controls whether Last -Modified headers are generated by the server.
<b>Syntax:</b>	SSILastModified on off
<b>Default:</b>	SSILastModified off
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.2.15 and later.

Under normal circumstances, a file filtered by `mod_include` may contain elements that are either dynamically generated, or that may have changed independently of the original file. As a result, by default the Last -Modified header is stripped from the response.

The `SSILastModified` directive overrides this behaviour, and allows the Last -Modified header to be respected if already present, or set if the header is not already present. This can be used to enable caching of the output. `SSILastModified` can take on the following values:

### **off**

The Last -Modified header will be stripped from responses, unless the `XBitHack` directive is set to `full` as described below.

### **on**

The Last -Modified header will be respected if already present in a response, and added to the response if the response is a file and the header is missing. The `SSILastModified` directive takes precedence over `XBitHack`.



## SSILegacyExprParser Directive

<b>Description:</b>	Enable compatibility mode for conditional expressions.
<b>Syntax:</b>	SSILegacyExprParser on off
<b>Default:</b>	SSILegacyExprParser off
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.3.13 and later.

As of version 2.3.13, `mod_include` has switched to the new `ap_expr` syntax for conditional expressions in `#if` flow control elements. This directive allows to switch to the `old syntax` which is compatible with Apache HTTPD version 2.2.x and earlier.



<b>Description:</b>	String that starts an include element
<b>Syntax:</b>	SSIStartTag <i>tag</i>
<b>Default:</b>	SSIStartTag "<! - -#"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_include

This directive changes the string that [mod\\_include](#) looks for to mark an include element to process.

You may want to use this option if you have 2 servers parsing the output of a file each processing different commands (possibly at different times).

```
SSIStartTag "<%"  
SSIEndTag   "%>"
```

The example given above, which also specifies a matching [SSIEndTag](#), will allow you to use SSI directives as shown in the example below:

### SSI directives with alternate start and end tags

```
<%printenv %>
```

## See also

- [SSIEndTag](#)



<b>Description:</b>	Configures the format in which date strings are displayed
<b>Syntax:</b>	SSITimeFormat <i>formatstring</i>
<b>Default:</b>	SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Base
<b>Module:</b>	mod_include

This directive changes the format in which date strings are displayed when echoing DATE environment variables. The *formatstring* is as in `strftime(3)` from the C standard library.

This directive has the same effect as the `<!--#config timefmt=formatstring -->` element.

```
SSITimeFormat "%R, %B %d, %Y"
```

The above directive would cause times to be displayed in the format "22:26, June 14, 2002".



<b>Description:</b>	String displayed when an unset variable is echoed
<b>Syntax:</b>	SSIUndefinedEcho <i>string</i>
<b>Default:</b>	SSIUndefinedEcho "(none)"
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Base
<b>Module:</b>	mod_include

This directive changes the string that `mod_include` displays when a variable is not set and "echoed".

```
SSIUndefinedEcho "<!-- undef -->"
```



<b>Description:</b>	Parse SSI directives in files with the execute bit set
<b>Syntax:</b>	XBitHack on off full
<b>Default:</b>	XBitHack off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_include

The **XBitHack** directive controls the parsing of ordinary html documents. This directive only affects files associated with the [MIME-type](#) text/html. **XBitHack** can take on the following values:

### **off**

No special treatment of executable files.

### **on**

Any text/html file that has the user-execute bit set will be treated as a server-parsed html document.

### **full**

As for on but also test the group-execute bit. If it is set, then set the Last-modified date of the returned file to be the last modified time of the file. If it is not set, then no last-modified date is sent. Setting this bit allows clients and proxies to cache the result of the request.

### **Note**

You would not want to use the full option, unless you assure the group-execute bit is unset for every SSI script which might `#include` a CGI or otherwise produces different output on each hit (or could potentially change on subsequent requests).

The [SSILastModified](#) directive takes precedence over the [XBitHack](#) directive when [SSILastModified](#) is set to on.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_info

```
┆  
┆  
┆ Extension  
┆ info_module  
┆ mod_info.c
```

mod\_info httpd.conf .

```
<Location /server-info>  
  SetHandler server-info  
</Location>
```

http://your.host.example.com/server-info



```
mod_info , ( , .htaccess)
```

```
, /,
```

```
mod_authz_host .
```

```
<Location /server-info>  
  SetHandler server-info  
  Order allow,deny  
  #  
  Allow from 127.0.0.1  
  # ,  
  Allow from 192.168.1.17  
</Location>
```



---

, (hook),

server-info . ,  
http://your.host.example.com/server-info?config

**?<module-name>**

**?config**

**?hooks**

(hook)

**?list**

**?server**



## mod\_info

- LoadModule, LoadFile [Server](#)
- Include, <IfModule>, <IfDefine>
- <Directory> , [mod\\_info](#)  
</Directory>
- mod\_perl



```
server-info
AddModuleInfo module-name string
,
Extension
mod_info
1.3
```

*module-name*            *string* HTML . ,

```
AddModuleInfo mod_deflate.c 'See <a \
href="http://www.apache.org/docs/2.4/mod/mod_deflate.html">\
http://www.apache.org/docs/docs/2.4/mod/mod_deflate.html</a>'
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_isapi

Windows ISAPI  
Extension

Base

isapi\_module

mod\_isapi.c

Win32 only

Internet Server extension API .  
Internet Server extension (, ISAPI .dll) .

Windows

ISAPI extension (.dll) .  
, . ISAPI extension ISAPI .

Apache Group



```
AddHandler ISAPI isapi-handler .  
.dll ISAPI extension httpd.conf .
```

```
AddHandler isapi-handler .dll
```

```
httpd.conf
```

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

```
ISAPI extension  
. , ISAPI .dll
```

```
ISAPI extension  
Options Ex
```

```
mod_isapi ISAPI
```

```
■ ■ .
```



---

ISAPI " " ISAPI 2.0 .  
ISAPI . ISA  
,  
ISAPILogNotSupported Off .

Microsoft IIS ISAPI extension

[ISAPICacheFile](#)

ISAPI extension . ,  
ISAPI

, ISAPI Extension , **ISAPI Filter** .

, .



2.0 `mod_isapi` , `ServerSupportFunction`

### **HSE\_REQ\_SEND\_URL\_REDIRECT\_RESP**

`URL` ( `http://server/location`).

### **HSE\_REQ\_SEND\_URL**

`URL` , `/location`

Microsoft `HSE_REQ_SEND_URL`

### **HSE\_REQ\_SEND\_RESPONSE\_HEADER**

`headers` ( `headers NULL` , `NULL` )

### **HSE\_REQ\_DONE\_WITH\_SESSION**

`ISAPI`

### **HSE\_REQ\_MAP\_URL\_TO\_PATH**

`()`

### **HSE\_APPEND\_LOG\_PARAMETER**

- `CustomLog` `\"%{isapi-parameter}n\"`
- `ISAPIAppendLogToQuery` `On` `%q`
- `ISAPIAppendLogToErrors` `On`  
`%{isapi-parameter}n`

### **HSE\_REQ\_IS\_KEEP\_CONN**

```

        Keep-Alive .
HSE_REQ_SEND_RESPONSE_HEADER_EX
        fKeepConn .
HSE_REQ_IS_CONNECTED
        false .

        ServerSupportFunction FALSE
        GetLastError ERROR_INVALID_PARAMETER .

        ReadClient ( ISAPIReadAheadBuffer )
        . ISAPIReadAheadBuffer (ISAPI )
        extension . , ISAPI extension Read(

        WriteClient , HSE_IO_SYNC ( 0 )
        . WriteClient FALSE , GetLastError
        ERROR_INVALID_PARAMETER .

        GetServerVariable , ( ) .
        GetServerVariable CGI ALL_HTTP

        2.0 mod_isapi ISAPI ,
        TransmitFile . , ISAPI .dll
        1.3 mod_isapi .

```



## ISAPIAppendLogToErrors

<a href="#">:</a>	ISAPI exntension	HSE_APPEND_LOG_PARAMETER
<a href="#">:</a>	ISAPIAppendLogToErrors	on off
<a href="#">:</a>	ISAPIAppendLogToErrors	off
<a href="#">:</a>		, , directory, .htaccess
<a href="#">Override :</a>	FileInfo	
<a href="#">:</a>	Base	
<a href="#">:</a>	mod_isapi	

ISAPI exntension HSE\_APPEND\_LOG\_PARAMETER

.



## ISAPIAppendLogToQuery

<a href="#">:</a>	ISAPI exntension	HSE_APPEND_LOG_PARAMETER
<a href="#">:</a>	ISAPIAppendLogToQuery	on off
<a href="#">:</a>	ISAPIAppendLogToQuery	on
<a href="#">:</a>		, , directory, .htaccess
<b>Override :</b>	FileInfo	
<a href="#">:</a>	Base	
<a href="#">:</a>	mod_isapi	

ISAPI exntension HSE\_APPEND\_LOG\_PARAMETER  
([CustomLog](#) %q ).



## ISAPI Extensions

```
:\ ISAPI.dll
:\ ISAPICacheFile file-path [file-path] ...
:\ ,
:\ Base
:\ mod_isapi
```

ServerRoot



```
ISAPI
ISAPIFakeAsync on|off
ISAPIFakeAsync off
, , directory, .htaccess
Override : FileInfo
Base
mod_isapi
```

on ISAPI .



ISAPI LogNotSupported

```
ISAPI extension
ISAPILogNotSupported on|off
ISAPILogNotSupported off
, , directory, .htaccess
Override : FileInfo
Base
mod_isapi
```

ISAPI extension .  
. ISAPI off .



```
ISAPI extension (read ahead
buffer)
ISAPIReadAheadBuffer size
ISAPIReadAheadBuffer 49152
, , directory, .htaccess
Override : FileInfo
Base
mod_isapi
```

ISAPI extension . ( )
ReadClient . ISAPI extension ReadCl
. ISAPI extension .



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_lbmethod_bybusyness`

<b>Description:</b>	Pending Request Counting load balancer scheduler algorithm for <a href="#">mod_proxy_balancer</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>lbmethod_bybusyness_module</code>
<b>Source File:</b>	<code>mod_lbmethod_bybusyness.c</code>
<b>Compatibility:</b>	Split off from <a href="#">mod_proxy_balancer</a> in 2.3

## Summary

This module does not provide any configuration directives of its own. It requires the services of [mod\\_proxy\\_balancer](#), and provides the `bybusyness` load balancing method.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[mod\\_proxy\\_balancer](#)



## Least-Request Counting Algorithm

Enabled via `lbmethod=bybusyness`, this scheduler keeps track of how many requests each worker is currently assigned at present. A new request is automatically assigned to the worker with the lowest number of active requests. This is useful in the case of workers that queue incoming requests independently of Apache, to ensure that queue length stays even and a request is always given to the worker most likely to service it the fastest and reduce latency.

In the case of multiple least-busy workers, the statistics (and weightings) used by the Request Counting method are used to break the tie. Over time, the distribution of work will come to resemble that characteristic of `byrequests` (as implemented by [mod\\_lbmethod\\_byrequests](#)).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_lbmethod_byrequests`

<b>Description:</b>	Request Counting load balancer scheduler algorithm for <a href="#">mod_proxy_balancer</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>lbmethod_byrequests_module</code>
<b>Source File:</b>	<code>mod_lbmethod_byrequests.c</code>
<b>Compatibility:</b>	Split off from <a href="#">mod_proxy_balancer</a> in 2.3

## Summary

This module does not provide any configuration directives of its own. It requires the services of [mod\\_proxy\\_balancer](#), and provides the `byrequests` load balancing method..



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[mod\\_proxy\\_balancer](#)



## Request Scheduling Algorithm

Enabled via `lbmethod=byrequests`, the idea behind this scheduler is that we distribute the requests among the various workers to ensure that each gets their configured share of the number of requests. It works as follows:

*lbfactor* is how much we expect this worker to work, or the workers' work quota. This is a normalized value representing their "share" of the amount of work to be done.

*lbstatus* is how urgent this worker has to work to fulfill its quota of work.

The *worker* is a member of the load balancer, usually a remote host serving one of the supported protocols.

We distribute each worker's work quota to the worker, and then look which of them needs to work most urgently (biggest *lbstatus*). This worker is then selected for work, and its *lbstatus* reduced by the total work quota we distributed to all workers. Thus the sum of all *lbstatus* does not change(\*) and we distribute the requests as desired.

If some workers are disabled, the others will still be scheduled correctly.

```
for each worker in workers
  worker lbstatus += worker lbfactor
  total factor    += worker lbfactor
  if worker lbstatus > candidate lbstatus
    candidate = worker

candidate lbstatus -= total factor
```

If a balancer is configured as follows:

<b>worker</b>	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
---------------	----------	----------	----------	----------

<b>lbfactor</b>	25	25	25	25
<b>lbstatus</b>	0	0	0	0

And *b* gets disabled, the following schedule is produced:

<b>worker</b>	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
<b>lbstatus</b>	-50	0	25	25
<b>lbstatus</b>	-25	0	-25	50
<b>lbstatus</b>	0	0	0	0
				(repeat)

That is it schedules: *a c d a c d a c d ...* Please note that:

<b>worker</b>	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
<b>lbfactor</b>	25	25	25	25

Has the exact same behavior as:

<b>worker</b>	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
<b>lbfactor</b>	1	1	1	1

This is because all values of *lbfactor* are normalized with respect to the others. For:

<b>worker</b>	<b>a</b>	<b>b</b>	<b>c</b>
<b>lbfactor</b>	1	4	1

worker *b* will, on average, get 4 times the requests that *a* and *c* will.

The following asymmetric configuration works as one would expect:

<b>worker</b>	<b>a</b>	<b>b</b>
	70	30

<b>lbfactor</b>		
<b>lbstatus</b>	-30	30
<b>lbstatus</b>	40	-40
<b>lbstatus</b>	10	-10
<b>lbstatus</b>	-20	20
<b>lbstatus</b>	-50	50
<b>lbstatus</b>	20	-20
<b>lbstatus</b>	-10	10
<b>lbstatus</b>	-40	40
<b>lbstatus</b>	30	-30
<b>lbstatus</b>	0	0
	(repeat)	

That is after 10 schedules, the schedule repeats and 7 *a* are selected with 3 *b* interspersed.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_lbmethod_bytraffic`

<b>Description:</b>	Weighted Traffic Counting load balancer scheduler algorithm for <a href="#">mod_proxy_balancer</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>lbmethod_bytraffic_module</code>
<b>Source File:</b>	<code>mod_lbmethod_bytraffic.c</code>
<b>Compatibility:</b>	Split off from <a href="#">mod_proxy_balancer</a> in 2.3

## Summary

This module does not provide any configuration directives of its own. It requires the services of [mod\\_proxy\\_balancer](#), and provides the `bytraffic` load balancing method..



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[mod\\_proxy\\_balancer](#)



## Weighted Request Counting Algorithm

Enabled via `lbmethod=bytraffic`, the idea behind this scheduler is very similar to the Request Counting method, with the following changes:

*lbfactor* is how much traffic, in bytes, we want this worker to handle. This is also a normalized value representing their "share" of the amount of work to be done, but instead of simply counting the number of requests, we take into account the amount of traffic this worker has either seen or produced.

If a balancer is configured as follows:

<b>worker</b>	<b>a</b>	<b>b</b>	<b>c</b>
<b>lbfactor</b>	1	2	1

Then we mean that we want *b* to process twice the amount of bytes than *a* or *c* should. It does not necessarily mean that *b* would handle twice as many requests, but it would process twice the I/O. Thus, the size of the request and response are applied to the weighting and selection algorithm.

Note: input and output bytes are weighted the same.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_lbmethod_heartbeat`

<b>Description:</b>	Heartbeat Traffic Counting load balancer scheduler algorithm for <a href="#">mod_proxy_balancer</a>
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>lbmethod_heartbeat_module</code>
<b>Source File:</b>	<code>mod_lbmethod_heartbeat.c</code>
<b>Compatibility:</b>	Available in version 2.3 and later

## Summary

`lbmethod=heartbeat` uses the services of [mod\\_heartmonitor](#) to balance between origin servers that are providing heartbeat info via the [mod\\_heartbeat](#) module.

This modules load balancing algorithm favors servers with more ready (idle) capacity over time, but does not select the server with the most ready capacity every time. Servers that have 0 active clients are penalized, with the assumption that they are not fully initialized.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[mod\\_proxy\\_balancer](#)

mod heartbeat  
mod heartmonitor



## HeartbeatStorage Directive

<b>Description:</b>	Path to read heartbeat data
<b>Syntax:</b>	HeartbeatStorage <i>file-path</i>
<b>Default:</b>	HeartbeatStorage logs/hb.dat
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lbmethod_heartbeat

The **HeartbeatStorage** directive specifies the path to read heartbeat data. This flat-file is used only when **mod\_slotmem\_shm** is not loaded.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_ldap

<b>Description:</b>	LDAP connection pooling and result caching services for use by other LDAP modules
<b>Status:</b>	Extension
<b>Module Identifier:</b>	ldap_module
<b>Source File:</b>	util_ldap.c

### Summary

This module was created to improve the performance of websites relying on backend connections to LDAP servers. In addition to the functions provided by the standard LDAP libraries, this module adds an LDAP connection pool and an LDAP shared memory cache.

To enable this module, LDAP support must be compiled into apr-util. This is achieved by adding the `--with-ldap` flag to the [configure](#) script when building Apache.

SSL/TLS support is dependent on which LDAP toolkit has been linked to [APR](#). As of this writing, APR-util supports: [OpenLDAP SDK](#) (2.x or later), [Novell LDAP SDK](#), [Mozilla LDAP SDK](#), native Solaris LDAP SDK (Mozilla based) or the native Microsoft LDAP SDK. See the [APR](#) website for details.



## Example Configuration

The following is an example configuration that uses [mod\\_ldap](#) to increase the performance of HTTP Basic authentication provided by [mod\\_authnz\\_ldap](#).

```
# Enable the LDAP connection pool and shared
# memory cache. Enable the LDAP cache status
# handler. Requires that mod_ldap and mod_a
# be loaded. Change the "yourdomain.example
# match your domain.

LDAPSharedCacheSize 500000
LDAPCacheEntries 1024
LDAPCacheTTL 600
LDAPOpCacheEntries 1024
LDAPOpCacheTTL 600

<Location "/ldap-status">
    SetHandler ldap-status

    Require host yourdomain.example.com

    Satisfy any
    AuthType Basic
    AuthName "LDAP Protected"
    AuthBasicProvider ldap
    AuthLDAPURL "ldap://127.0.0.1/dc=example
    Require valid-user
</Location>
```



## LDAP CONNECTION POOL

LDAP connections are pooled from request to request. This allows the LDAP server to remain connected and bound ready for the next request, without the need to unbind/connect/rebind. The performance advantages are similar to the effect of HTTP keepalives.

On a busy server it is possible that many requests will try and access the same LDAP server connection simultaneously. Where an LDAP connection is in use, Apache will create a new connection alongside the original one. This ensures that the connection pool does not become a bottleneck.

There is no need to manually enable connection pooling in the Apache configuration. Any module using this module for access to LDAP services will share the connection pool.

LDAP connections can keep track of the ldap client credentials used when binding to an LDAP server. These credentials can be provided to LDAP servers that do not allow anonymous binds during referral chasing. To control this feature, see the [LDAPReferrals](#) and [LDAPReferralHopLimit](#) directives. By default, this feature is enabled.



For improved performance, [mod\\_ldap](#) uses an aggressive caching strategy to minimize the number of times that the LDAP server must be contacted. Caching can easily double or triple the throughput of Apache when it is serving pages protected with [mod\\_authnz\\_ldap](#). In addition, the load on the LDAP server will be significantly decreased.

[mod\\_ldap](#) supports two types of LDAP caching during the search/bind phase with a *search/bind cache* and during the compare phase with two *operation caches*. Each LDAP URL that is used by the server has its own set of these three caches.

## The Search/Bind Cache

The process of doing a search and then a bind is the most time-consuming aspect of LDAP operation, especially if the directory is large. The search/bind cache is used to cache all searches that resulted in successful binds. Negative results (*i.e.*, unsuccessful searches, or searches that did not result in a successful bind) are not cached. The rationale behind this decision is that connections with invalid credentials are only a tiny percentage of the total number of connections, so by not caching invalid credentials, the size of the cache is reduced.

[mod\\_ldap](#) stores the username, the DN retrieved, the password used to bind, and the time of the bind in the cache. Whenever a new connection is initiated with the same username, [mod\\_ldap](#) compares the password of the new connection with the password in the cache. If the passwords match, and if the cached entry is not too old, [mod\\_ldap](#) bypasses the search/bind phase.

The search and bind cache is controlled with the [LDAPCacheEntries](#) and [LDAPCacheTTL](#) directives.

## Operation Caches

During attribute and distinguished name comparison functions, [mod\\_ldap](#) uses two operation caches to cache the compare operations. The first compare cache is used to cache the results of compares done to test for LDAP group membership. The second compare cache is used to cache the results of comparisons done between distinguished names.

Note that, when group membership is being checked, any sub-group comparison results are cached to speed future sub-group comparisons.

The behavior of both of these caches is controlled with the [LDAPOpCacheEntries](#) and [LDAPOpCacheTTL](#) directives.

## Monitoring the Cache

[mod\\_ldap](#) has a content handler that allows administrators to monitor the cache performance. The name of the content handler is `ldap-status`, so the following directives could be used to access the [mod\\_ldap](#) cache information:

```
<Location "/server/cache-info">  
    SetHandler ldap-status  
</Location>
```

By fetching the URL `http://servername/cache-info`, the administrator can get a status report of every cache that is used by [mod\\_ldap](#) cache. Note that if Apache does not support shared memory, then each [httpd](#) instance has its own cache, so reloading the URL will result in different information each time, depending on which [httpd](#) instance processes the request.



The ability to create an SSL and TLS connections to an LDAP server is defined by the directives [LDAPTrustedGlobalCert](#), [LDAPTrustedClientCert](#) and [LDAPTrustedMode](#). These directives specify the CA and optional client certificates to be used, as well as the type of encryption to be used on the connection (none, SSL or TLS/STARTTLS).

```
# Establish an SSL LDAP connection on port 636
# mod_ldap and mod_authnz_ldap be loaded. Cf
# "yourdomain.example.com" to match your domain

LDAPTrustedGlobalCert CA_DER "/certs/certifi

<Location "/ldap-status">
    SetHandler ldap-status

    Require host yourdomain.example.com

    Satisfy any
    AuthType Basic
    AuthName "LDAP Protected"
    AuthBasicProvider ldap
    AuthLDAPURL "ldaps://127.0.0.1/dc=example.com"
    Require valid-user
</Location>
```

```
# Establish a TLS LDAP connection on port 389
# mod_ldap and mod_authnz_ldap be loaded. Cf
# "yourdomain.example.com" to match your domain

LDAPTrustedGlobalCert CA_DER "/certs/certifi

<Location "/ldap-status">
    SetHandler ldap-status
```

```
Require host yourdomain.example.com
```

```
Satisfy any
```

```
AuthType Basic
```

```
AuthName "LDAP Protected"
```

```
AuthBasicProvider ldap
```

```
AuthLDAPURL "ldap://127.0.0.1/dc=example"
```

```
Require valid-user
```

```
</Location>
```



The different LDAP SDKs have widely different methods of setting and handling both CA and client side certificates.

If you intend to use SSL or TLS, read this section CAREFULLY so as to understand the differences between configurations on the different LDAP toolkits supported.

## **Netscape/Mozilla/iPlanet SDK**

CA certificates are specified within a file called cert7.db. The SDK will not talk to any LDAP server whose certificate was not signed by a CA specified in this file. If client certificates are required, an optional key3.db file may be specified with an optional password. The secmod file can be specified if required. These files are in the same format as used by the Netscape Communicator or Mozilla web browsers. The easiest way to obtain these files is to grab them from your browser installation.

Client certificates are specified per connection using the LDAPTrustedClientCert directive by referring to the certificate "nickname". An optional password may be specified to unlock the certificate's private key.

The SDK supports SSL only. An attempt to use STARTTLS will cause an error when an attempt is made to contact the LDAP server at runtime.

```
# Specify a Netscape CA certificate file
LDAPTrustedGlobalCert CA_CERT7_DB "/certs/ca
# Specify an optional key3.db file for client
LDAPTrustedGlobalCert CERT_KEY3_DB "/certs/l
# Specify the secmod file if required
LDAPTrustedGlobalCert CA_SECMOD "/certs/secr
<Location "/ldap-status">
    SetHandler ldap-status
```

```
Require host yourdomain.example.com

Satisfy any
AuthType Basic
AuthName "LDAP Protected"
AuthBasicProvider ldap
LDAPTrustedClientCert CERT_NICKNAME <nic
AuthLDAPURL "ldaps://127.0.0.1/dc=examp
Require valid-user
</Location>
```

## Novell SDK

One or more CA certificates must be specified for the Novell SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.

Note: Client certificates are specified globally rather than per connection, and so must be specified with the `LDAPTrustedGlobalCert` directive as below. Trying to set client certificates via the `LDAPTrustedClientCert` directive will cause an error to be logged when an attempt is made to connect to the LDAP server..

The SDK supports both SSL and STARTTLS, set using the `LDAPTrustedMode` parameter. If an `ldaps://` URL is specified, SSL mode is forced, override this directive.

```
# Specify two CA certificate files
LDAPTrustedGlobalCert CA_DER "/certs/cacert
LDAPTrustedGlobalCert CA_BASE64 "/certs/cac
# Specify a client certificate file and key
LDAPTrustedGlobalCert CERT_BASE64 "/certs/ce
LDAPTrustedGlobalCert KEY_BASE64 "/certs/key
```

```
# Do not use this directive, as it will thro  
#LDAPTrustedClientCert CERT_BASE64 "/certs/c
```

## OpenLDAP SDK

One or more CA certificates must be specified for the OpenLDAP SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.

Both CA and client certificates may be specified globally (LDAPTrustedGlobalCert) or per-connection (LDAPTrustedClientCert). When any settings are specified per-connection, the global settings are superseded.

The documentation for the SDK claims to support both SSL and STARTTLS, however STARTTLS does not seem to work on all versions of the SDK. The SSL/TLS mode can be set using the LDAPTrustedMode parameter. If an ldaps:// URL is specified, SSL mode is forced. The OpenLDAP documentation notes that SSL (ldaps://) support has been deprecated to be replaced with TLS, although the SSL functionality still works.

```
# Specify two CA certificate files  
LDAPTrustedGlobalCert CA_DER "/certs/cacert:  
LDAPTrustedGlobalCert CA_BASE64 "/certs/cac  
<Location "/ldap-status">  
    SetHandler ldap-status  
  
Require host yourdomain.example.com  
  
LDAPTrustedClientCert CERT_BASE64 "/cert  
LDAPTrustedClientCert KEY_BASE64 "/certs  
# CA certs respecified due to per-direct  
LDAPTrustedClientCert CA_DER "/certs/cac  
LDAPTrustedClientCert CA_BASE64 "/certs,
```

```
Satisfy any
AuthType Basic
AuthName "LDAP Protected"
AuthBasicProvider ldap
AuthLDAPURL "ldaps://127.0.0.1/dc=example"
Require valid-user
</Location>
```

## Solaris SDK

SSL/TLS for the native Solaris LDAP libraries is not yet supported. If required, install and use the OpenLDAP libraries instead.

## Microsoft SDK

SSL/TLS certificate configuration for the native Microsoft LDAP libraries is done inside the system registry, and no configuration directives are required.

Both SSL and TLS are supported by using the ldaps:// URL format, or by using the LDAPTrustedMode directive accordingly.

Note: The status of support for client certificates is not yet known for this toolkit.



<b>Description:</b>	Maximum number of entries in the primary LDAP cache
<b>Syntax:</b>	LDAPCacheEntries <i>number</i>
<b>Default:</b>	LDAPCacheEntries 1024
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Specifies the maximum size of the primary LDAP cache. This cache contains successful search/binds. Set it to 0 to turn off search/bind caching. The default size is 1024 cached searches.



<b>Description:</b>	Time that cached items remain valid
<b>Syntax:</b>	LDAPCacheTTL <i>seconds</i>
<b>Default:</b>	LDAPCacheTTL 600
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Specifies the time (in seconds) that an item in the search/bind cache remains valid. The default is 600 seconds (10 minutes).



<b>Description:</b>	Discard backend connections that have been sitting in the connection pool too long
<b>Syntax:</b>	<code>LDAPConnectionPoolTTL n</code>
<b>Default:</b>	<code>LDAPConnectionPoolTTL -1</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ldap</code>
<b>Compatibility:</b>	Apache HTTP Server 2.3.12 and later

Specifies the maximum age, in seconds, that a pooled LDAP connection can remain idle and still be available for use. Connections are cleaned up when they are next needed, not asynchronously.

A setting of 0 causes connections to never be saved in the backend connection pool. The default value of -1, and any other negative value, allows connections of any age to be reused.

For performance reasons, the reference time used by this directive is based on when the LDAP connection is returned to the pool, not the time of the last successful I/O with the LDAP server.

Since 2.4.10, new measures are in place to avoid the reference time from being inflated by cache hits or slow requests. First, the reference time is not updated if no backend LDAP connections were needed. Second, the reference time uses the time the HTTP request was received instead of the time the request is completed.

This timeout defaults to units of seconds, but accepts suffixes for milliseconds (ms), minutes (min), and hours (h).



<b>Description:</b>	Specifies the socket connection timeout in seconds
<b>Syntax:</b>	<code>LDAPConnectionTimeout</code> <i>seconds</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ldap</code>

This directive configures the `LDAP_OPT_NETWORK_TIMEOUT` (or `LDAP_OPT_CONNECT_TIMEOUT`) option in the underlying LDAP client library, when available. This value typically controls how long the LDAP client library will wait for the TCP connection to the LDAP server to complete.

If a connection is not successful with the timeout period, either an error will be returned or the LDAP client library will attempt to connect to a secondary LDAP server if one is specified (via a space-separated list of hostnames in the [AuthLDAPURL](#)).

The default is 10 seconds, if the LDAP client library linked with the server supports the `LDAP_OPT_NETWORK_TIMEOUT` option.

`LDAPConnectionTimeout` is only available when the LDAP client library linked with the server supports the `LDAP_OPT_NETWORK_TIMEOUT` (or `LDAP_OPT_CONNECT_TIMEOUT`) option, and the ultimate behavior is dictated entirely by the LDAP client library.



## LDAPLibraryDebug Directive

<b>Description:</b>	Enable debugging in the LDAP SDK
<b>Syntax:</b>	LDAPLibraryDebug 7
<b>Default:</b>	disabled
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Turns on SDK-specific LDAP debug options that generally cause the LDAP SDK to log verbose trace information to the main Apache error log. The trace messages from the LDAP SDK provide gory details that can be useful during debugging of connectivity problems with backend LDAP servers

This option is only configurable when Apache HTTP Server is linked with an LDAP SDK that implements LDAP\_OPT\_DEBUG or LDAP\_OPT\_DEBUG\_LEVEL, such as OpenLDAP (a value of 7 is verbose) or Tivoli Directory Server (a value of 65535 is verbose).

The logged information will likely contain plaintext credentials being used or validated by LDAP authentication, so care should be taken in protecting and purging the error log when this directive is used.



**Description:** Number of entries used to cache LDAP compare operations

**Syntax:** LDAPOpCacheEntries *number*

**Default:** LDAPOpCacheEntries 1024

**Context:** server config

**Status:** Extension

**Module:** mod\_ldap

This specifies the number of entries [mod\\_ldap](#) will use to cache LDAP compare operations. The default is 1024 entries. Setting it to 0 disables operation caching.



<b>Description:</b>	Time that entries in the operation cache remain valid
<b>Syntax:</b>	LDAPOpCacheTTL <i>seconds</i>
<b>Default:</b>	LDAPOpCacheTTL 600
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Specifies the time (in seconds) that entries in the operation cache remain valid. The default is 600 seconds.



**Description:** The maximum number of referral hops to chase before terminating an LDAP query.

**Syntax:** LDAPReferralHopLimit *number*

**Default:** SDK dependent, typically between 5 and 10

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod\_ldap

This directive, if enabled by the `LDAPReferrals` directive, limits the number of referral hops that are followed before terminating an LDAP query.

Support for this tunable is uncommon in LDAP SDKs.



<b>Description:</b>	Enable referral chasing during queries to the LDAP server.
<b>Syntax:</b>	<code>LDAPReferrals On Off default</code>
<b>Default:</b>	<code>LDAPReferrals On</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap
<b>Compatibility:</b>	The <i>default</i> parameter is available in Apache 2.4.7 and later

Some LDAP servers divide their directory among multiple domains and use referrals to direct a client when a domain boundary is crossed. This is similar to a HTTP redirect. LDAP client libraries may or may not chase referrals by default. This directive explicitly configures the referral chasing in the underlying SDK.

`LDAPReferrals` takes the following values:

#### **"on"**

When set to "on", the underlying SDK's referral chasing state is enabled, `LDAPReferralHopLimit` is used to override the SDK's hop limit, and an LDAP rebind callback is registered.

#### **"off"**

When set to "off", the underlying SDK's referral chasing state is disabled completely.

#### **"default"**

When set to "default", the underlying SDK's referral chasing state is not changed, `LDAPReferralHopLimit` is not used to override the SDK's hop limit, and no LDAP rebind callback is registered.

The directive `LDAPReferralHopLimit` works in conjunction with this directive to limit the number of referral hops to follow before terminating the LDAP query. When referral processing is enabled by a value of "On", client credentials will be provided, via a rebind callback, for any LDAP server requiring them.



<b>Description:</b>	Configures the number of LDAP server retries.
<b>Syntax:</b>	LDAPRetries <i>number-of-retries</i>
<b>Default:</b>	LDAPRetries 3
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

The server will retry failed LDAP requests up to **LDAPRetries** times. Setting this directive to 0 disables retries.

LDAP errors such as timeouts and refused connections are retryable.



## LDAPRetryDelay Directive

<b>Description:</b>	Configures the delay between LDAP server retries.
<b>Syntax:</b>	LDAPRetryDelay <i>seconds</i>
<b>Default:</b>	LDAPRetryDelay 0
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

If **LDAPRetryDelay** is set to a non-zero value, the server will delay retrying an LDAP request for the specified amount of time. Setting this directive to 0 will result in any retry to occur without delay.

LDAP errors such as timeouts and refused connections are retryable.



<b>Description:</b>	Sets the shared memory cache file
<b>Syntax:</b>	LDAPSharedCacheFile <i>directory-path/filename</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Specifies the directory path and file name of the shared memory cache file. If not set, anonymous shared memory will be used if the platform supports it.



<b>Description:</b>	Size in bytes of the shared-memory cache
<b>Syntax:</b>	LDAPSharedCacheSize <i>bytes</i>
<b>Default:</b>	LDAPSharedCacheSize 500000
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Specifies the number of bytes to allocate for the shared memory cache. The default is 500kb. If set to 0, shared memory caching will not be used and every HTTPD process will create its own cache.



<b>Description:</b>	Specifies the timeout for LDAP search and bind operations, in seconds
<b>Syntax:</b>	LDAPTimeout <i>seconds</i>
<b>Default:</b>	LDAPTimeout 60
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap
<b>Compatibility:</b>	Apache HTTP Server 2.3.5 and later

This directive configures the timeout for bind and search operations, as well as the LDAP\_OPT\_TIMEOUT option in the underlying LDAP client library, when available.

If the timeout expires, httpd will retry in case an existing connection has been silently dropped by a firewall. However, performance will be much better if the firewall is configured to send TCP RST packets instead of silently dropping packets.

Timeouts for ldap compare operations requires an SDK with LDAP\_OPT\_TIMEOUT, such as OpenLDAP >= 2.4.4.



<b>Description:</b>	Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates.
<b>Syntax:</b>	<code>LDAPTrustedClientCert type directory-path/filename/nickname [password]</code>
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

It specifies the directory path, file name or nickname of a per connection client certificate used when establishing an SSL or TLS connection to an LDAP server. Different locations or directories may have their own independent client certificate settings. Some LDAP toolkits (notably Novell) do not support per connection client certificates, and will throw an error on LDAP server connection if you try to use this directive (Use the LDAPTrustedGlobalCert directive instead for Novell client certificates - See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- CA\_DER - binary DER encoded CA certificate
- CA\_BASE64 - PEM encoded CA certificate
- CERT\_DER - binary DER encoded client certificate
- CERT\_BASE64 - PEM encoded client certificate
- CERT\_NICKNAME - Client certificate "nickname" (Netscape SDK)
- KEY\_DER - binary DER encoded private key
- KEY\_BASE64 - PEM encoded private key



<b>Description:</b>	Sets the file or database containing global trusted Certificate Authority or global client certificates
<b>Syntax:</b>	LDAPTrustedGlobalCert <i>type directory-path/filename [password]</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

It specifies the directory path and file name of the trusted CA certificates and/or system wide client certificates `mod_ldap` should use when establishing an SSL or TLS connection to an LDAP server. Note that all certificate information specified using this directive is applied globally to the entire server installation. Some LDAP toolkits (notably Novell) require all client certificates to be set globally using this directive. Most other toolkits require clients certificates to be set per Directory or per Location using LDAPTrustedClientCert. If you get this wrong, an error may be logged when an attempt is made to contact the LDAP server, or the connection may silently fail (See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- CA\_DER - binary DER encoded CA certificate
- CA\_BASE64 - PEM encoded CA certificate
- CA\_CERT7\_DB - Netscape cert7.db CA certificate database file
- CA\_SECMOD - Netscape secmod database file
- CERT\_DER - binary DER encoded client certificate
- CERT\_BASE64 - PEM encoded client certificate
- CERT\_KEY3\_DB - Netscape key3.db client certificate database file
- CERT\_NICKNAME - Client certificate "nickname" (Netscape

SDK)

- CERT\_PFX - PKCS#12 encoded client certificate (Novell SDK)
- KEY\_DER - binary DER encoded private key
- KEY\_BASE64 - PEM encoded private key
- KEY\_PFX - PKCS#12 encoded private key (Novell SDK)



**Description:** Specifies the SSL/TLS mode to be used when connecting to an LDAP server.

**Syntax:** LDAPTrustedMode *type*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_ldap

The following modes are supported:

- NONE - no encryption
- SSL - ldaps:// encryption on default port 636
- TLS - STARTTLS encryption on default port 389

Not all LDAP toolkits support all the above modes. An error message will be logged at runtime if a mode is not supported, and the connection to the LDAP server will fail.

If an ldaps:// URL is specified, the mode becomes SSL and the setting of LDAPTrustedMode is ignored.



<b>Description:</b>	Force server certificate verification
<b>Syntax:</b>	LDAPVerifyServerCert <i>On Off</i>
<b>Default:</b>	LDAPVerifyServerCert On
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

Specifies whether to force the verification of a server certificate when establishing an SSL connection to the LDAP server.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_log\_config

```
├──  
├── Base  
├── log_config_module  
└── mod_log_config.c
```

[CustomLog](#) , [LogFormat](#) ,  
[TransferLog](#) , [TransferLog CustomL](#)



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)



## LogFormat CustomLog

C "\n" "\t"

" %" .

%%	
%. . . a	IP-
%. . . A	() IP-
%. . . B	HTTP .
%. . . b	HTTP . CLF 0 ' -' .
%. . . {Foobar}C	<i>Foobar</i> .
%. . . D	( ).
%. . . {FOOBAR}e	<i>FOOBAR</i>
%. . . f	
%. . . h	
%. . . H	
%. . . {Foobar}i	<i>Foobar</i> : .
%. . . l	( identd ) . <u>mod_ident</u> <u>IdentityCheck On</u> .
%. . . m	
%. . . {Foobar}n	<i>Foobar</i> (note) .
%. . . {Foobar}o	<i>Foobar</i> : .

%...p	
%...P	ID.
%... {format}P	ID ID. format
%...q	( ? , )
%...r	
%...s	(status). ** %...>s.
%...t	common log format ( )
%... {format}t	strftime(3) format.( )
%...T	( ).
%...u	(auth , ( %s) 401 )
%...U	URL .
%...v	<u>ServerName</u> .
%...V	<u>UseCanonicalName</u> .
%...X	. <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> X = .  + = (keep  alive).  - = . </div> ( 1.3 %...c, ssl {var}c .)
%...I	0 . .
%...O	0 . <span style="float: right;"><u>mod</u></span>

```
"..." ( , "%h %u %r %s %b" ) , (
  "-"). "!" HTTP
"%!400,501{User-agent}i" 400 (Bad Request) 501 (Not
Implemented) User-agent : ,
"%!200,304,302{Referer}i" Referer :
```

```
"<" ">"
%T, %D, %r , % . %>
(status) , %<u .
```

```
2.0.46 httpd 2.0 %...r,%...i,%...o .
Common Log Format . ,
```

```
2.0.46 \xhh . hh
" \, C (
```

### Common Log Format (CLF)

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

### Common Log Format

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

### NCSA extended/combined

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-agent}i\""
```

### Referer

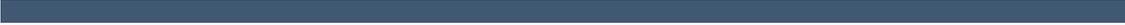
```
"%{Referer}i -> %U"
```

### Agent ()

```
"%{User-agent}i"
```

```
ServerName Listen %v %p .
```





- [: Buffer log entries in memory before writing to disk](#)
- [:](#)
- [:](#)
- [: Base](#)
- [: mod\\_log\\_config](#)

Documentation not yet translated. Please see English version of document.



## CustomLog

```
CustomLog file|pipe format|nickname [env=[!]environment-variable]
```

**CustomLog** . ,

### **file**

ServerRoot .

### **pipe**

" |"

```
root
```

LogFormat

*format* .

```
# CustomLog
```

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

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

```
( 'env=!name' )
```

```
mod_setenvif mod_rewrite
```

```
GIF
```

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



- [\[:\]](#) Sets filename and format of log file
- [\[:\]](#) `GlobalLogfile|pipe format|nickname [env=[!]environment-variable| expr=expression]`
- [\[:\]](#)
- [\[:\]](#) Base
- [\[:\]](#) mod\_log\_config
- [\[:\]](#) Available in Apache HTTP Server 2.4.19 and later

The documentation for this directive has not been translated yet.  
Please have a look at the English version.



## LogFormat

```
LogFormat format|nickname [nickname]
LogFormat "%h %l %u %t \"%r\" %>s %b"
Base
mod_log_config
```

```
LogFormat format
LogFormat ( )
LogFormat format nickname
LogFormat CustomLog
LogFormat LogFormat Transfe
LogFormat ( %)
```

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



## TransferLog

```
:\n:\nTransferLog file|pipe\n:\n,\n:\nBase\n:\nmod_log_config
```

CustomLog ( ) LogFormat  
Common Log Format .

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



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_log_debug`

<b><u>Description:</u></b>	Additional configurable debug logging
<b><u>Status:</u></b>	Experimental
<b><u>Module Identifier:</u></b>	<code>log_debug_module</code>
<b><u>Source File:</u></b>	<code>mod_log_debug.c</code>
<b><u>Compatibility:</u></b>	Available in Apache 2.3.14 and later



1. Log message after request to `/foo/*` is processed:

```
<Location "/foo/">
  LogMessage "/foo/ has been requested"
</Location>
```

2. Log message if request to `/foo/*` is processed in a sub-request:

```
<Location "/foo/">
  LogMessage "subrequest to /foo/" hook=
</Location>
```

The default `log_transaction` hook is not executed for sub-requests, therefore we have to use a different hook.

3. Log message if an IPv6 client causes a request timeout:

```
LogMessage "IPv6 timeout from %{REMOTE_A
```

Note the placing of the double quotes for the `expr=` argument.

4. Log the value of the "X-Foo" request environment variable in each stage of the request:

```
<Location "/">
  LogMessage "%{reqenv:X-Foo}" hook=all
</Location>
```

Together with microsecond time stamps in the error log, `hook=all` also lets you determine the times spent in the

different parts of the request processing.



## LogMessage Directive

<b>Description:</b>	Log user-defined message to error log
<b>Syntax:</b>	LogMessage <i>message</i> [hook= <i>hook</i> ] [expr= <i>expression</i> ]
<b>Default:</b>	Unset
<b>Context:</b>	directory
<b>Status:</b>	Experimental
<b>Module:</b>	mod_log_debug

This directive causes a user defined message to be logged to the error log. The message can use variables and functions from the [ap\\_expr syntax](#). References to HTTP headers will not cause header names to be added to the Vary header. The messages are logged at loglevel info.

The hook specifies before which phase of request processing the message will be logged. The following hooks are supported:

Name
translate_name
type_checker
quick_handler
map_to_storage
check_access
check_access_ex
insert_filter
check_authn
check_authz
fixups
handler
log_transaction

The default is `log_transaction`. The special value `all` is also supported, causing a message to be logged at each phase. Not all hooks are executed for every request.

The optional expression allows to restrict the message if a condition is met. The details of the expression syntax are described in the [ap\\_expr documentation](#). References to HTTP headers will not cause the header names to be added to the Vary header.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_log\_forensic

<b>Description:</b>	Forensic Logging of the requests made to the server
<b>Status:</b>	Extension
<b>Module Identifier:</b>	log_forensic_module
<b>Source File:</b>	mod_log_forensic.c
<b>Compatibility:</b>	<a href="#">mod_unique_id</a> is no longer required since version 2.1

### Summary

This module provides for forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request. The forensic logger is very strict, which means:

- The format is fixed. You cannot modify the logging format at runtime.
- If it cannot write its data, the child process exits immediately and may dump core (depending on your [CoreDumpDirectory](#) configuration).

The `check_forensic` script, which can be found in the distribution's support directory, may be helpful in evaluating the forensic log output.



### Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Apache Log Files](#)

[mod\\_log\\_config](#)



## Forensic Log Format

Each request is logged two times. The first time is *before* it's processed further (that is, after receiving the headers). The second log entry is written *after* the request processing at the same time where normal logging occurs.

In order to identify each request, a unique request ID is assigned. This forensic ID can be cross logged in the normal transfer log using the `%{forensic-id}` format string. If you're using [mod\\_unique\\_id](#), its generated ID will be used.

The first line logs the forensic ID, the request line and all received headers, separated by pipe characters (`|`). A sample line looks like the following (all on one line):

```
+yQtJf8CoAB4AAFNBIEAAAAA|GET /manual/de/images/down.gif
HTTP/1.1|Host:localhost%3a8080|User-Agent:Mozilla/5.0 (X11; U;
Linux i686; en-US; rv%3a1.6) Gecko/20040216
Firefox/0.8|Accept:image/png, etc...
```

The plus character at the beginning indicates that this is the first log line of this request. The second line just contains a minus character and the ID again:

```
-yQtJf8CoAB4AAFNBIEAAAAA
```

The `check_forensic` script takes as its argument the name of the logfile. It looks for those +/- ID pairs and complains if a request was not completed.



See the [security tips](#) 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.

The log files may contain sensitive data such as the contents of `Authorization:` headers (which can contain passwords), so they should not be readable by anyone except the user that starts the server.



## ForensicLog Directive

<b>Description:</b>	Sets filename of the forensic log
<b>Syntax:</b>	ForensicLog <i>filename pipe</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_log_forensic

The **ForensicLog** directive is used to log requests to the server for forensic analysis. Each log entry is assigned a unique ID which can be associated with the request using the normal **CustomLog** directive. **mod\_log\_forensic** creates a token called `forensic-id`, which can be added to the transfer log using the `%{forensic-id}n` format string.

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

### ***filename***

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. The program name can be specified relative to the **ServerRoot** directive.

### **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 or switches to a less privileged user.

### **Note**

When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashes 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.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_logio

- 
- Extension
- logio\_module
- mod\_logio.c

SSL/TLS , SSL/TLS

mod\_log\_config.



## Bugfix checklist

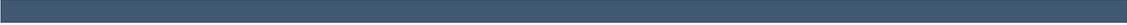
[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

mod\_log\_config





. " %" :  
:

%. . . I	. 0 .
%. . . 0	. 0 .

:

:

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"  
\"%{User-agent}i\" %I %0"
```



## LogIOTrackTTFB

<a href="#">:</a>	Enable tracking of time to first byte (TTFB)
<a href="#">:</a>	LogIOTrackTTFB ON OFF
<a href="#">:</a>	LogIOTrackTTFB OFF
<a href="#">:</a>	, , directory, .htaccess
<a href="#">Override :</a>	none
<a href="#">:</a>	Extension
<a href="#">:</a>	mod_logio
<a href="#">:</a>	Apache HTTP Server 2.4.13 and later

The documentation for this directive has not been translated yet.  
Please have a look at the [English version](#).



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_lua

<b>Description:</b>	Provides Lua hooks into various portions of the httpd request processing
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	lua_module
<b>Source File:</b>	mod_lua.c
<b>Compatibility:</b>	2.3 and later

### Summary

This module allows the server to be extended with scripts written in the Lua programming language. The extension points (hooks) available with [mod\\_lua](#) include many of the hooks available to natively compiled Apache HTTP Server modules, such as mapping requests to files, generating dynamic responses, access control, authentication, and authorization

More information on the Lua programming language can be found at the [the Lua website](#).

mod\_lua is still in experimental state. Until it is declared stable, usage and behavior may change at any time, even between stable releases of the 2.4.x series. Be sure to check the CHANGES file before upgrading.

### Warning

This module holds a great deal of power over httpd, which is both a strength and a potential security risk. It is **not** recommended that you use this module on a server that is shared with users you do not trust, as it can be abused to change the internal workings of httpd.



## Basic Configuration

The basic module loading directive is

```
LoadModule lua_module modules/mod_lua.so
```

mod\_lua provides a handler named lua-script, which can be used with a [SetHandler](#) or [AddHandler](#) directive:

```
<Files "*.lua">  
    SetHandler lua-script  
</Files>
```

This will cause mod\_lua to handle requests for files ending in .lua by invoking that file's handle function.

For more flexibility, see [LuaMapHandler](#).



In the Apache HTTP Server API, the handler is a specific kind of hook responsible for generating the response. Examples of modules that include a handler are [mod\\_proxy](#), [mod\\_cgi](#), and [mod\\_status](#).

`mod_lua` always looks to invoke a Lua function for the handler, rather than just evaluating a script body CGI style. A handler function looks something like this:

```
example.lua
-- example handler

require "string"

--[[
    This is the default method name for Lua
    function-name in the LuaMapHandler dir
    entry point.
--]]
function handle(r)
    r.content_type = "text/plain"

    if r.method == 'GET' then
        r:puts("Hello Lua World!\n")
        for k, v in pairs( r:parseargs() ) do
            r:puts( string.format("%s: %s\n", k, v) )
        end
    elseif r.method == 'POST' then
        r:puts("Hello Lua World!\n")
        for k, v in pairs( r:parsebody() ) do
            r:puts( string.format("%s: %s\n", k, v) )
        end
    elseif r.method == 'PUT' then
        -- use our own Error contents
        r:puts("Unsupported HTTP method " .. r.method)
    end
end
```

```
        r.status = 405
        return apache2.0K
    else
    -- use the ErrorDocument
        return 501
    end
    return apache2.0K
end
```

This handler function just prints out the uri or form encoded arguments to a plaintext page.

This means (and in fact encourages) that you can have multiple handlers (or hooks, or filters) in the same script.



**Setting Authorization Provider**

`mod_authz_core` provides a high-level interface to authorization that is much easier to use than using into the relevant hooks directly. The first argument to the `Require` directive gives the name of the responsible authorization provider. For any `Require` line, `mod_authz_core` will call the authorization provider of the given name, passing the rest of the line as parameters. The provider will then check authorization and pass the result as return value.

The authz provider is normally called before authentication. If it needs to know the authenticated user name (or if the user will be authenticated at all), the provider must return `apache2.AUTHZ_DENIED_NO_USER`. This will cause authentication to proceed and the authz provider to be called a second time.

The following authz provider function takes two arguments, one ip address and one user name. It will allow access from the given ip address without authentication, or if the authenticated user matches the second argument:

```
authz_provider.lua  
  
require 'apache2'  
  
function authz_check_foo(r, ip, user)  
    if r.useragent_ip == ip then  
        return apache2.AUTHZ_GRANTED  
    elseif r.user == nil then  
        return apache2.AUTHZ_DENIED_NO_USER  
    elseif r.user == user then  
        return apache2.AUTHZ_GRANTED  
    else
```

```
        return apache2.AUTHZ_DENIED
    end
end
```

The following configuration registers this function as provider foo and configures it for URL /:

```
LuaAuthzProvider foo authz_provider.lua authz_provider.lua
<Location "/">
    Require foo 10.1.2.3 john_doe
</Location>
```



## Hook Functions

Hook functions are how modules (and Lua scripts) participate in the processing of requests. Each type of hook exposed by the server exists for a specific purpose, such as mapping requests to the file system, performing access control, or setting mime types:

Hook phase	mod_lua directive	Description
Quick handler	<a href="#">LuaQuickHandler</a>	This is the first hook that will be called after a request has been mapped to a host or virtual host
Translate name	<a href="#">LuaHookTranslateName</a>	This phase translates the requested URI into a filename on the system. Modules such as <a href="#">mod_alias</a> and <a href="#">mod_rewrite</a> operate in this phase.
Map to storage	<a href="#">LuaHookMapToStorage</a>	This phase maps files to their physical, cached or external/proxied storage. It can be used by proxy or caching modules
Check Access	<a href="#">LuaHookAccessChecker</a>	This phase checks whether a client has access to a resource. This phase is run before the user is authenticated, so beware.

Check User ID	<a href="#"><u>LuaHookCheckUserID</u></a>	This phase it used to check the negotiated user ID
Check Authorization	<a href="#"><u>LuaHookAuthChecker</u></a> or <a href="#"><u>LuaAuthzProvider</u></a>	This phase authorizes a user based on the negotiated credentials, such as user ID, client certificate etc.
Check Type	<a href="#"><u>LuaHookTypeChecker</u></a>	This phase checks the requested file and assigns a content type and a handler to it
Fixups	<a href="#"><u>LuaHookFixups</u></a>	This is the final "fix anything" phase before the content handlers are run. Any last-minute changes to the request should be made here.
Content handler	fx. <code>.lua</code> files or through <a href="#"><u>LuaMapHandler</u></a>	This is where the content is handled. Files are read, parsed, some are run, and the result is sent to the client
Logging	<a href="#"><u>LuaHookLog</u></a>	Once a request has been handled, it enters several logging phases, which logs the request in either the error or access log. <code>Mod_lua</code> is able to hook into the start of

		this and control logging output.
--	--	----------------------------------

Hook functions are passed the request object as their only argument (except for `LuaAuthzProvider`, which also gets passed the arguments from the `Require` directive). They can return any value, depending on the hook, but most commonly they'll return `OK`, `DONE`, or `DECLINED`, which you can write in Lua as `apache2.OK`, `apache2.DONE`, or `apache2.DECLINED`, or else an HTTP status code.

#### **translate\_name.lua**

```
-- example hook that rewrites the URI to a  
require 'apache2'  
  
function translate_name(r)  
    if r.uri == "/translate-name" then  
        r.filename = r.document_root .. "/f  
        return apache2.OK  
    end  
    -- we don't care about this URL, give a  
    return apache2.DECLINED  
end
```

#### **translate\_name2.lua**

```
--[[ example hook that rewrites one URI to  
apache2.DECLINED to give other URL map  
substitution, including the core trans  
on the DocumentRoot.
```

Note: Use the `early/late` flags in the  
or after `mod_alias`.

```
--]]  
  
require 'apache2'  
  
function translate_name(r)  
    if r.uri == "/translate-name" then  
        r.uri = "/find_me.txt"  
        return apache2.DECLINED  
    end  
    return apache2.DECLINED  
end
```



## request\_rec

The request\_rec is mapped in as a userdata. It has a metatable which lets you do useful things with it. For the most part it has the same fields as the request\_rec struct, many of which are writable as well as readable. (The table fields' content can be changed, but the fields themselves cannot be set to different tables.)

Name	Lua type	Writable	Description
allowoverrides	string	no	The AllowOverride to the current
ap_auth_type	string	no	If an authentication made, this is authentication
args	string	yes	The query string extracted from foo=bar&na
assbackwards	boolean	no	Set to true if (style request no headers) )
auth_name	string	no	The realm name authorization
banner	string	no	The server banner HTTP Server openssl/0.
basic_auth_pw	string	no	The basic authentication this request,
canonical_filename	string	no	The canonical request
content_encoding	string	no	The content encoding

			current request
content_type	string	yes	The content type of the request, as determined by the <code>type_check</code> plugin. For example, <code>image/gif</code> .
context_prefix	string	no	
context_document_root	string	no	
document_root	string	no	The document root of the request.
err_headers_out	table	no	MIME headers to be added to the response, prior to sending. This table can be used to persist a redirect.
filename	string	yes	The file name to be served. For example, <code>/www/example.com</code> can be changed to <code>/www/example.com/</code> to map a request to a handler (or serve a different file) or to serve a different file than the one requested.
handler	string	yes	The name of the handler to serve. For example, <code>lua-script</code> by <code>mod_lua</code> . See the <a href="#">AddHandler</a> directives, but <code>mod_lua</code> to add a handler to serve up a file that would otherwise be served.
headers_in	table	yes	MIME headers to be added to the request.

			the request. Fields such as Host, Referer and
headers_out	table	yes	MIME headers in response.
hostname	string	no	The host name. Host: header
is_https	boolean	no	Whether or not done via HTTPS
is_initial_req	boolean	no	Whether this is a request or a response
limit_req_body	number	no	The size limit for this request
log_id	string	no	The ID to identify access and error
method	string	no	The request method. POST.
notes	table	yes	A list of notes on from one or more
options	string	no	The Options for the current request
path_info	string	no	The PATH_INFO for this request.
port	number	no	The server port for this request.
protocol	string	no	The protocol used for this request
proxyreq	string	yes	Denotes whether request or not, generally set to post_read_request phase of a request

range	string	no	The contents header.
remaining	number	no	The number of bytes to be read from the request.
server_built	string	no	The time the server was built.
server_name	string	no	The server name.
some_auth_required	boolean	no	Whether some authentication is/was required for this request.
subprocess_env	table	yes	The environment variables for this request.
started	number	no	The time the server (re)started, in epoch (Jan 1 1970).
status	number	yes	The (current) status of this request, as defined in RFC 2616.
the_request	string	no	The request line, including client, f.x. GET / HTTP/1.1.
unparsed_uri	string	no	The unparsed part of the URI.
uri	string	yes	The URI after parsing by httpd.
user	string	yes	If an authentication has been made, the user name of the authentication.
useragent_ip	string	no	The IP of the client of the request.



The `request_rec` object has (at least) the following methods:

```
r:flush()    -- flushes the output buffer.
              -- Returns true if the flush wa

while we_have_stuff_to_send do
    r:puts("Bla bla bla\n") -- print someth
    r:flush() -- flush the buffer (send to
    r.usleep(500000) -- fake processing tim
end
```

```
r:addoutputfilter(name|function) -- add an
r:addoutputfilter("fooFilter") -- add the f
```

```
r:sendfile(filename) -- sends an entire fil

if use_sendfile_thing then
    r:sendfile("/var/www/large_file.img")
end
```

```
r:parseargs() -- returns two tables; one st
              -- and one for multi-value da

local GET, GETMULTI = r:parseargs()
r:puts("Your name is: " .. GET['name'] or "
```

```
r:parsebody([sizeLimit]) -- parse the reque
                          -- just like r:par
                          -- An optional num
```

```
        -- of bytes to par  
local POST, POSTMULTI = r:parsebody(1024*10  
r:puts("Your name is: " .. POST['name'] or
```

```
r:puts("hello", " world", "!") -- print to
```

```
r:write("a single string") -- print to resp
```

```
r:escape_html("<html>test</html>") -- Escap
```

```
r:base64_encode(string) -- Encodes a string  
local encoded = r:base64_encode("This is a
```

```
r:base64_decode(string) -- Decodes a Base64  
local decoded = r:base64_decode("VGhpcyBpcy
```

```
r:md5(string) -- Calculates and returns the  
local hash = r:md5("This is a test") -- ret
```

```
r:sha1(string) -- Calculates and returns th  
local hash = r:sha1("This is a test") -- re
```

```
r:escape(string) -- URL-Escapes a string:
```

```
local url = "http://foo.bar/1 2 3 & 4 + 5"  
local escaped = r:escape(url) -- returns 'h
```

```
r:unescape(string) -- Unescapes an URL-escape
```

```
local url = "http%3a%2f%2ffoo.bar%2f1+2+3+%  
local unescaped = r:unescape(url) -- return
```

```
r:construct_url(string) -- Constructs an UR
```

```
local url = r:construct_url(r.uri)
```

```
r.mpm_query(number) -- Queries the server f
```

```
local mpm = r.mpm_query(14)  
if mpm == 1 then  
    r:puts("This server uses the Event MPM")  
end
```

```
r:expr(string) -- Evaluates an expr string.
```

```
if r:expr("%{HTTP_HOST} =~ /^www/") then  
    r:puts("This host name starts with www")  
end
```

```
r:scoreboard_process(a) -- Queries the serv
```

```
local process = r:scoreboard_process(1)
r:puts("Server 1 has PID " .. process.pid)
```

```
r:scoreboard_worker(a, b) -- Queries for in

local thread = r:scoreboard_worker(1, 1)
r:puts("Server 1's thread 1 has thread ID "
```

```
r:clock() -- Returns the current time with
```

```
r:requestbody(filename) -- Reads and return
-- If 'filename' is specific
-- contents to that file:
```

```
local input = r:requestbody()
r:puts("You sent the following request body
r:puts(input)
```

```
r:add_input_filter(filter_name) -- Adds 'fi
```

```
r.module_info(module_name) -- Queries the s
```

```
local mod = r.module_info("mod_lua.c")
if mod then
    for k, v in pairs(mod.commands) do
        r:puts( ("%s: %s\n"):format(k,v) ) --
    end
end
```

```
r:loaded_modules() -- Returns a list of mod  
  
for k, module in pairs(r:loaded_modules())  
  r:puts("I have loaded module " .. module)  
end
```

```
r:runtime_dir_relative(filename) -- Compute  
  -- relative to the
```

```
r:server_info() -- Returns a table containi  
  -- the name of the httpd ex
```

```
r:set_document_root(file_path) -- Sets the
```

```
r:set_context_info(prefix, docroot) -- Sets
```

```
r:os_escape_path(file_path) -- Converts an
```

```
r:escape_logitem(string) -- Escapes a string
```

```
r.strcmp_match(string, pattern) -- Checks if  
  -- fx. whether 'www
```

```
local match = r.strcmp_match("foobar.com",  
if match then  
  r:puts("foobar.com matches foo*.com")  
end
```

```
r:set_keepalive() -- Sets the keepalive sta
```

```
r:make_etag() -- Constructs and returns the
```

```
r:send_interim_response(clear) -- Sends an  
-- if 'clear' is true
```

```
r:custom_response(status_code, string) -- C  
-- This work
```

```
r:custom_response(404, "Baleted!")
```

```
r.exists_config_define(string) -- Checks wh  
  
if r.exists_config_define("FOO") then  
    r:puts("httpd was probably run with -DF  
end
```

```
r:state_query(string) -- Queries the server
```

```
r:stat(filename [,wanted]) -- Runs stat() o  
  
local info = r:stat("/var/www/foo.txt")  
if info then  
    r:puts("This file exists and was last m  
end
```

```
r:regex(string, pattern [,flags]) -- Runs a
local matches = r:regex("foo bar baz", [[fo
if matches then
    r:puts("The regex matched, and the last
end

-- Example ignoring case sensitivity:
local matches = r:regex("FOO bar BAz", [[(f

-- Flags can be a bitwise combination of:
-- 0x01: Ignore case
-- 0x02: Multiline search
```

```
r.usleep(number_of_microseconds) -- Puts th
```

```
r:dbacquire(dbType[, dbParams]) -- Acquires
-- See 'Database co
```

```
r:ivm_set("key", value) -- Set an Inter-VM
-- These values per
-- and so should on
-- Values can be nu
-- per process basi

r:ivm_get("key") -- Fetches a variab
-- if it exists or

-- An example getter/setter that saves a gl
function handle(r)
    -- First VM to call this will get no va
    local foo = r:ivm_get("cached_data")
```

```
    if not foo then
        foo = do_some_calcs() -- fake some
        r:ivm_set("cached_data", foo) -- se
    end
    r:puts("Cached data is: ", foo)
end
```

```
r:htpassword(string [,algorithm [,cost]]) -
```

```
r:mkdir(dir [,mode]) -- Creates a directory
```

```
r:mkrdir(dir [,mode]) -- Creates directorie
```

```
r:rmdir(dir) -- Removes a directory.
```

```
r:touch(file [,mtime]) -- Sets the file mod
```

```
r:get_direnties(dir) -- Returns a table wi
```

```
function handle(r)
    local dir = r.context_document_root
    for _, f in ipairs(r:get_direnties(dir))
        local info = r:stat(dir .. "/" .. f)
        if info then
            local mtime = os.date(fmt, info.mtime)
            local ftype = (info.filetype == 2) and
            r:puts( ("%s %s %10i %s\n"):format(ft
```

```
    end
  end
end
```

```
r.date_parse_rfc(string) -- Parses a date/t
```

```
r:getcookie(key) -- Gets a HTTP cookie
```

```
r:setcookie{
  key = [key],
  value = [value],
  expires = [expiry],
  secure = [boolean],
  httponly = [boolean],
  path = [path],
  domain = [domain]
} -- Sets a HTTP cookie, for instance:
```

```
r:setcookie{
  key = "cookie1",
  value = "HDHfa9eyffh396rt",
  expires = os.time() + 86400,
  secure = true
}
```

```
r:wsupgrade() -- Upgrades a connection to W
if r:wsupgrade() then -- if we can upgrade:
  r:wswrite("Welcome to websockets!") --
  r:wsclose() -- goodbye!
end
```

```
r:wsread() -- Reads a WebSocket frame from  
  
local line, isFinal = r:wsread() -- isFinal  
                                -- If it is  
r:wswrite("You wrote: " .. line)
```

```
r:wswrite(line) -- Writes a frame to a WebS  
r:wswrite("Hello, world!")
```

```
r:wsclose() -- Closes a WebSocket request a  
  
if r:wsupgrade() then  
    r:wswrite("Write something: ")  
    local line = r:wsread() or "nothing"  
    r:wswrite("You wrote: " .. line);  
    r:wswrite("Goodbye!")  
    r:wsclose()  
end
```



## Logging Functions

```
-- examples of logging messages
r:trace1("This is a trace log message") --
r:debug("This is a debug log message")
r:info("This is an info log message")
r:notice("This is a notice log message")
r:warn("This is a warn log message")
r:err("This is an err log message")
r:alert("This is an alert log message")
r:crit("This is a crit log message")
r:emerg("This is an emerg log message")
```



A package named `apache2` is available with (at least) the following contents.

**`apache2.OK`**

internal constant `OK`. Handlers should return this if they've handled the request.

**`apache2.DECLINED`**

internal constant `DECLINED`. Handlers should return this if they are not going to handle the request.

**`apache2.DONE`**

internal constant `DONE`.

**`apache2.version`**

Apache HTTP server version string

**`apache2.HTTP_MOVED_TEMPORARILY`**

HTTP status code

**`apache2.PROXYREQ_NONE`, `apache2.PROXYREQ_PROXY`,  
`apache2.PROXYREQ_REVERSE`,  
`apache2.PROXYREQ_RESPONSE`**

internal constants used by [mod\\_proxy](#)

**`apache2.AUTHZ_DENIED`, `apache2.AUTHZ_GRANTED`,  
`apache2.AUTHZ_NEUTRAL`,  
`apache2.AUTHZ_GENERAL_ERROR`,  
`apache2.AUTHZ_DENIED_NO_USER`**

internal constants used by [mod\\_authz\\_core](#)

(Other HTTP status codes are not yet implemented.)



Filter functions implemented via [LuaInputFilter](#) or [LuaOutputFilter](#) are designed as three-stage non-blocking functions using coroutines to suspend and resume a function as buckets are sent down the filter chain. The core structure of such a function is:

```
function filter(r)
    -- Our first yield is to signal that we
    -- Before this yield, we can set up our
    -- and, if we deem it necessary, decline
    if something_bad then
        return -- This would skip this filter
    end
    -- Regardless of whether we have data to
    -- Note that only output filters can pro
    -- final stage to append data to the co
    coroutine.yield([optional header to be

    -- After we have yielded, buckets will
    -- do whatever we want with them and th
    -- Buckets are stored in the global var
    -- that checks if 'bucket' is not nil:
    while bucket ~= nil do
        local output = mangle(bucket) -- Do
        coroutine.yield(output) -- Return o
    end

    -- Once the buckets are gone, 'bucket'
    -- loop and land us here. Anything extr
    -- can be done by doing a final yield h
    -- can append data to the content in th
    coroutine.yield([optional footer to be

end
```



Mod\_lua implements a simple database feature for querying and running commands on the most popular database engines (mySQL, PostgreSQL, FreeTDS, ODBC, SQLite, Oracle) as well as mod\_dbd.

The example below shows how to acquire a database handle and return information from a table:

```
function handle(r)
  -- Acquire a database handle
  local database, err = r:dbacquire("mysq
  if not err then
    -- Select some information from it
    local results, err = database:selec
    if not err then
      local rows = results(0) -- fetc
      for k, row in pairs(rows) do
        r:puts( string.format("Name
      end
    else
      r:puts("Database query error: "
    end
    database:close()
  else
    r:puts("Could not connect to the da
  end
end
```

To utilize [mod\\_dbd](#), specify mod\_dbd as the database type, or leave the field blank:

```
local database = r:dbacquire("mod_dbd")
```

## Database object and contained functions

The database object returned by `dbacquire` has the following methods:

### Normal select and query from a database:

```
-- Run a statement and return the number of
local affected, errmsg = database:query(r,

-- Run a statement and return a result set
local result, errmsg = database:select(r, "
```

### Using prepared statements (recommended):

```
-- Create and run a prepared statement:
local statement, errmsg = database:prepare(
if not errmsg then
    local result, errmsg = statement:query(
end

-- Fetch a prepared statement from a DBDPre
local statement, errmsg = database:prepared
if not errmsg then
    local result, errmsg = statement:select
end
```

### Escaping values, closing databases etc:

```
-- Escape a value for use in a statement:
local escaped = database:escape(r, ["'|bla

-- Close a database connection and free up
database:close()
```

```
-- Check whether a database connection is u  
local connected = database:active()
```

## Working with result sets

The result set returned by `db:select` or by the prepared statement functions created through `db:prepare` can be used to fetch rows synchronously or asynchronously, depending on the row number specified:

`result(0)` fetches all rows in a synchronous manner, returning a table of rows.

`result(-1)` fetches the next available row in the set, asynchronously.

`result(N)` fetches row number N, asynchronously:

```
-- fetch a result set using a regular query  
local result, err = db:select(r, "SELECT *  
  
local rows = result(0) -- Fetch ALL rows sy  
local row = result(-1) -- Fetch the next av  
local row = result(1234) -- Fetch row numbe  
local row = result(-1, true) -- Fetch the n
```

One can construct a function that returns an iterative function to iterate over all rows in a synchronous or asynchronous way, depending on the `async` argument:

```
function rows(resultset, async)  
  local a = 0  
  local function getnext()  
    a = a + 1  
    local row = resultset(-1)
```

```

        return row and a or nil, row
    end
    if not async then
        return pairs(resultset(0))
    else
        return getnext, self
    end
end
end

local statement, err = db:prepare(r, "SELEC
if not err then
    -- fetch rows asynchronously:
    local result, err = statement:select(20
    if not err then
        for index, row in rows(result, true
            ....
        end
    end
end

    -- fetch rows synchronously:
    local result, err = statement:select(20
    if not err then
        for index, row in rows(result, fals
            ....
        end
    end
end
end
end

```

## Closing a database connection

Database handles should be closed using `database:close()` when they are no longer needed. If you do not close them manually, they will eventually be garbage collected and closed by `mod_lua`, but you may end up having too many unused connections to the database if you leave the closing up to `mod_lua`. Essentially, the following two measures are the same:

```
-- Method 1: Manually close a handle
local database = r:dbacquire("mod_dbd")
database:close() -- All done

-- Method 2: Letting the garbage collector
local database = r:dbacquire("mod_dbd")
database = nil -- throw away the reference
collectgarbage() -- close the handle via GC
```

## Precautions when working with databases

Although the standard query and run functions are freely available, it is recommended that you use prepared statements whenever possible, to both optimize performance (if your db handle lives on for a long time) and to minimize the risk of SQL injection attacks. run and query should only be used when there are no variables inserted into a statement (a static statement). When using dynamic statements, use db:prepare or db:prepared.



**Description:** Plug an authorization provider function into [mod\\_authz\\_core](#)

**Syntax:** LuaAuthzProvider provider\_name  
/path/to/lua/script.lua  
function\_name

**Context:** server config

**Status:** Experimental

**Module:** mod\_lua

**Compatibility:** 2.4.3 and later

After a lua function has been registered as authorization provider, it can be used with the [Require](#) directive:

```
LuaRoot "/usr/local/apache2/lua"  
LuaAuthzProvider foo authz.lua authz_check_foo  
<Location "/">  
  Require foo johndoe  
</Location>
```

```
require "apache2"  
function authz_check_foo(r, who)  
  if r.user ~= who then return apache2.AUTHZ_DENIED  
  return apache2.AUTHZ_GRANTED  
end
```



<b>Description:</b>	Configure the compiled code cache.
<b>Syntax:</b>	<code>LuaCodeCache stat   forever   never</code>
<b>Default:</b>	<code>LuaCodeCache stat</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_lua</code>

Specify the behavior of the in-memory code cache. The default is `stat`, which stats the top level script (not any included ones) each time that file is needed, and reloads it if the modified time indicates it is newer than the one it has already loaded. The other values cause it to keep the file cached forever (don't stat and replace) or to never cache the file.

In general `stat` or `forever` is good for production, and `stat` or `never` for development.

### Examples:

```
LuaCodeCache stat
LuaCodeCache forever
LuaCodeCache never
```



<b>Description:</b>	Provide a hook for the <code>access_checker</code> phase of request processing
<b>Syntax:</b>	<code>LuaHookAccessChecker</code> <code>/path/to/lua/script.lua</code> <code>hook_function_name [early late]</code>
<b>Context:</b>	server config, virtual host, directory, <code>.htaccess</code>
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_lua</code>
<b>Compatibility:</b>	The optional third argument is supported in 2.3.15 and later

Add your hook to the `access_checker` phase. An access checker hook function usually returns `OK`, `DECLINED`, or `HTTP_FORBIDDEN`.

### Ordering

The optional arguments "early" or "late" control when this script runs relative to other modules.



<b>Description:</b>	Provide a hook for the auth_checker phase of request processing
<b>Syntax:</b>	LuaHookAuthChecker /path/to/lua/script.lua hook_function_name [early late]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua
<b>Compatibility:</b>	The optional third argument is supported in 2.3.15 and later

Invoke a lua function in the auth\_checker phase of processing a request. This can be used to implement arbitrary authentication and authorization checking. A very simple example:

```
require 'apache2'

-- fake authcheck hook
-- If request has no auth info, set the res
-- return a 401 to ask the browser for basi
-- If request has auth info, don't actually
-- pretend we got userid 'foo' and validate
-- Then check if the userid is 'foo' and ac
function authcheck_hook(r)

    -- look for auth info
    auth = r.headers_in['Authorization']
    if auth ~= nil then
        -- fake the user
        r.user = 'foo'
    end
end
```

```
if r.user == nil then
  r:debug("authcheck: user is nil, retu
  r.err_headers_out['WWW-Authenticate']
  return 401
elseif r.user == "foo" then
  r:debug('user foo: OK')
else
  r:debug("authcheck: user='" .. r.user
  r.err_headers_out['WWW-Authenticate']
  return 401
end
return apache2.OK
end
```

## Ordering

The optional arguments "early" or "late" control when this script runs relative to other modules.



<b>Description:</b>	Provide a hook for the check_user_id phase of request processing
<b>Syntax:</b>	LuaHookCheckUserID /path/to/lua/script.lua hook_function_name [early late]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua
<b>Compatibility:</b>	The optional third argument is supported in 2.3.15 and later

...

### Ordering

The optional arguments "early" or "late" control when this script runs relative to other modules.



## LuaHookFixups Directive

<b>Description:</b>	Provide a hook for the fixups phase of a request processing
<b>Syntax:</b>	<code>LuaHookFixups /path/to/lua/script.lua hook_function_name</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

Just like `LuaHookTranslateName`, but executed at the fixups phase



<b>Description:</b>	Provide a hook for the insert_filter phase of request processing
<b>Syntax:</b>	LuaHookInsertFilter /path/to/lua/script.lua hook_function_name
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

Not Yet Implemented



<b>Description:</b>	Provide a hook for the access log phase of a request processing
<b>Syntax:</b>	<code>LuaHookLog /path/to/lua/script.lua log_function_name</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

This simple logging hook allows you to run a function when httpd enters the logging phase of a request. With it, you can append data to your own logs, manipulate data before the regular log is written, or prevent a log entry from being created. To prevent the usual logging from happening, simply return `apache2 . DONE` in your logging handler, otherwise return `apache2 . OK` to tell httpd to log as normal.

Example:

```
LuaHookLog "/path/to/script.lua" logger
```

```
-- /path/to/script.lua --
function logger(r)
  -- flip a coin:
  -- If 1, then we write to our own Lua log
  -- in the main log.
  -- If 2, then we just sanitize the output
  -- log the sanitized bits.

  if math.random(1,2) == 1 then
    -- Log stuff ourselves and don't log
    local f = io.open("/foo/secret.log")
    if f then
```

```
        f:write("Something secret happened")
        f:close()
    end
    return apache2.DONE -- Tell httpd no more data
else
    r.uri = r.uri:gsub("somesecretstuff", "secret")
    return apache2.OK -- tell httpd to read more
end
end
```



**Description:** Provide a hook for the map\_to\_storage phase of request processing

**Syntax:** LuaHookMapToStorage  
/path/to/lua/script.lua  
hook\_function\_name

**Context:** server config, virtual host, directory, .htaccess

**Override:** All

**Status:** Experimental

**Module:** mod\_lua

Like [LuaHookTranslateName](#) but executed at the map-to-storage phase of a request. Modules like mod\_cache run at this phase, which makes for an interesting example on what to do here:

```
LuaHookMapToStorage "/path/to/lua/script.lua"
```

```
require"apache2"  
cached_files = {}  
  
function read_file(filename)  
    local input = io.open(filename, "r")  
    if input then  
        local data = input:read("*a")  
        cached_files[filename] = data  
        file = cached_files[filename]  
        input:close()  
    end  
    return cached_files[filename]  
end  
  
function check_cache(r)
```

```
if r.filename:match("%.png$") then -- 0
    local file = cached_files[r.filename]
    if not file then
        file = read_file(r.filename)
    end
    if file then -- If file exists, write it
        r.status = 200
        r:write(file)
        r:info("Sent %s to client from cache")
        return apache2.DONE -- skip default handler
    end
end
return apache2.DECLINED -- If we had no handler
```



<b>Description:</b>	Provide a hook for the translate name phase of request processing
<b>Syntax:</b>	<code>LuaHookTranslateName</code> <code>/path/to/lua/script.lua</code> <code>hook_function_name [early late]</code>
<b>Context:</b>	server config, virtual host
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_lua</code>
<b>Compatibility:</b>	The optional third argument is supported in 2.3.15 and later

Add a hook (at `APR_HOOK_MIDDLE`) to the translate name phase of request processing. The hook function receives a single argument, the `request_rec`, and should return a status code, which is either an HTTP error code, or the constants defined in the `apache2` module: `apache2.OK`, `apache2.DECLINED`, or `apache2.DONE`.

For those new to hooks, basically each hook will be invoked until one of them returns `apache2.OK`. If your hook doesn't want to do the translation it should just return `apache2.DECLINED`. If the request should stop processing, then return `apache2.DONE`.

Example:

```
# httpd.conf
LuaHookTranslateName "/scripts/conf/hooks.lua"
```

```
-- /scripts/conf/hooks.lua --
require "apache2"
function silly_mapper(r)
```

```
if r.uri == "/" then
    r.filename = "/var/www/home.lua"
    return apache2.OK
else
    return apache2.DECLINED
end
end
```

### Context

This directive is not valid in [<Directory>](#), [<Files>](#), or htaccess context.

### Ordering

The optional arguments "early" or "late" control when this script runs relative to other modules.



**Description:** Provide a hook for the `type_checker` phase of request processing

**Syntax:** `LuaHookTypeChecker`  
`/path/to/lua/script.lua`  
`hook_function_name`

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** All

**Status:** Experimental

**Module:** `mod_lua`

This directive provides a hook for the `type_checker` phase of the request processing. This phase is where requests are assigned a content type and a handler, and thus can be used to modify the type and handler based on input:

```
LuaHookTypeChecker "/path/to/lua/script.lua"
```

```
function type_checker(r)
    if r.uri:match("%.to_gif$") then --
        r.content_type = "image/gif" --
        r.handler = "gifWizard" --
        r.filename = r.uri:gsub("%.to_g
        return apache2.OK
    end

    return apache2.DECLINED
end
```



<b>Description:</b>	Controls how parent configuration sections are merged into children
<b>Syntax:</b>	<code>LuaInherit none parent-first parent-last</code>
<b>Default:</b>	<code>LuaInherit parent-first</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_lua</code>
<b>Compatibility:</b>	2.4.0 and later

By default, if LuaHook\* directives are used in overlapping Directory or Location configuration sections, the scripts defined in the more specific section are run *after* those defined in the more generic section (LuaInherit parent-first). You can reverse this order, or make the parent context not apply at all.

In previous 2.3.x releases, the default was effectively to ignore LuaHook\* directives from parent configuration sections.



<b>Description:</b>	Provide a Lua function for content input filtering
<b>Syntax:</b>	<pre> LuaInputFilter filter_name /path/to/lua/script.lua function_name </pre>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua
<b>Compatibility:</b>	2.4.5 and later

Provides a means of adding a Lua function as an input filter. As with output filters, input filters work as coroutines, first yielding before buffers are sent, then yielding whenever a bucket needs to be passed down the chain, and finally (optionally) yielding anything that needs to be appended to the input data. The global variable `bucket` holds the buckets as they are passed onto the Lua script:

```

LuaInputFilter myInputFilter "/www/filter.lua"
<Files "*.lua">
  SetInputFilter myInputFilter
</Files>

```

```

--[[
  Example input filter that converts all
]]--
function input_filter(r)
  print("luaInputFilter called") -- debug
  coroutine.yield() -- Yield and wait for
  while bucket do -- For each bucket, do.
    local output = string.upper(bucket)
    coroutine.yield(output) -- Send con
  end
end

```

```
-- No more buckets available.  
coroutine.yield("&filterSignature=1234"  
end
```

The input filter supports denying/skipping a filter if it is deemed unwanted:

```
function input_filter(r)  
  if not good then  
    return -- Simply deny filtering, pa  
  end  
  coroutine.yield() -- wait for buckets  
  ... -- insert filter stuff here  
end
```

See "[Modifying contents with Lua filters](#)" for more information.



## lua-map-handler-directive

<b>Description:</b>	Map a path to a lua handler
<b>Syntax:</b>	<code>LuaMapHandler uri-pattern /path/to/lua/script.lua [function-name]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

This directive matches a uri pattern to invoke a specific handler function in a specific file. It uses PCRE regular expressions to match the uri, and supports interpolating match groups into both the file path and the function name. Be careful writing your regular expressions to avoid security issues.

### Examples:

```
LuaMapHandler "/(\w+)/(\w+)" "/scripts/$1.lua" "handle_$2"
```

This would match uri's such as `/photos/show?id=9` to the file `/scripts/photos.lua` and invoke the handler function `handle_show` on the lua vm after loading that file.

```
LuaMapHandler "/bingo" "/scripts/wombat.lua"
```

This would invoke the "handle" function, which is the default if no specific function name is provided.



<b>Description:</b>	Provide a Lua function for content output filtering
<b>Syntax:</b>	<pre>LuaOutputFilter filter_name /path/to/lua/script.lua function_name</pre>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua
<b>Compatibility:</b>	2.4.5 and later

Provides a means of adding a Lua function as an output filter. As with input filters, output filters work as coroutines, first yielding before buffers are sent, then yielding whenever a bucket needs to be passed down the chain, and finally (optionally) yielding anything that needs to be appended to the input data. The global variable `bucket` holds the buckets as they are passed onto the Lua script:

```
LuaOutputFilter myOutputFilter "/www/filter
<Files "*.lua">
  SetOutputFilter myOutputFilter
</Files>
```

```
--[[
  Example output filter that escapes all
]]--
function output_filter(r)
  coroutine.yield("(Handled by myOutputFi

  while bucket do -- For each bucket, do.
    local output = r:escape_html(bucket
    coroutine.yield(output) -- Send con
  end
```

```
-- No more buckets available.  
end
```

As with the input filter, the output filter supports denying/skipping a filter if it is deemed unwanted:

```
function output_filter(r)  
  if not r.content_type:match("text/html")  
    return -- Simply deny filtering, pa  
  end  
  coroutine.yield() -- wait for buckets  
  ... -- insert filter stuff here  
end
```

### Lua filters with `mod_filter`

When a Lua filter is used as the underlying provider via the `FilterProvider` directive, filtering will only work when the *filter-name* is identical to the *provider-name*.

See "[Modifying contents with Lua filters](#)" for more information.



## LuaPackageCPath Directive

<b>Description:</b>	Add a directory to lua's package.cpath
<b>Syntax:</b>	LuaPackageCPath /path/to/include/? .soa
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

Add a path to lua's shared library search path. Follows the same conventions as lua. This just munges the package.cpath in the lua vms.



## lua\_package\_path Directives

<b>Description:</b>	Add a directory to lua's package.path
<b>Syntax:</b>	LuaPackagePath /path/to/include/?.lua
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

Add a path to lua's module search path. Follows the same conventions as lua. This just munges the package.path in the lua vms.

### Examples:

```
LuaPackagePath "/scripts/lib/?.lua"  
LuaPackagePath "/scripts/lib/?.lua/init.lua"
```



**Description:** Provide a hook for the quick handler of request processing

**Syntax:** `LuaQuickHandler /path/to/script.lua hook_function_name`

**Context:** server config, virtual host

**Override:** All

**Status:** Experimental

**Module:** mod\_lua

This phase is run immediately after the request has been mapped to a virtual host, and can be used to either do some request processing before the other phases kick in, or to serve a request without the need to translate, map to storage et cetera. As this phase is run before anything else, directives such as [<Location>](#) or [<Directory>](#) are void in this phase, just as URIs have not been properly parsed yet.

### Context

This directive is not valid in [<Directory>](#), [<Files>](#), or htaccess context.



<b>Description:</b>	Specify the base path for resolving relative paths for mod_lua directives
<b>Syntax:</b>	LuaRoot /path/to/a/directory
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

Specify the base path which will be used to evaluate all relative paths within mod\_lua. If not specified they will be resolved relative to the current working directory, which may not always work well for a server.



<b>Description:</b>	One of once, request, conn, thread -- default is once
<b>Syntax:</b>	LuaScope once   request   conn   thread   server [min] [max]
<b>Default:</b>	LuaScope once
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Experimental
<b>Module:</b>	mod_lua

Specify the life cycle scope of the Lua interpreter which will be used by handlers in this "Directory." The default is "once"

**once:**

use the interpreter once and throw it away.

**request:**

use the interpreter to handle anything based on the same file within this request, which is also request scoped.

**conn:**

Same as request but attached to the connection\_rec

**thread:**

Use the interpreter for the lifetime of the thread handling the request (only available with threaded MPMs).

**server:**

This one is different than others because the server scope is quite long lived, and multiple threads will have the same server\_rec. To accommodate this, server scoped Lua states are stored in an apr resource list. The min and max arguments specify the minimum and maximum number of Lua states to keep in the pool.

Generally speaking, the thread and server scopes execute roughly 2-3 times faster than the rest, because they don't have to spawn new Lua states on every request (especially with the event MPM, as even keepalive requests will use a new thread for each request). If you are satisfied that your scripts will not have problems reusing a state, then the thread or server scopes should be used for maximum performance. While the thread scope will provide the fastest responses, the server scope will use less memory, as states are pooled, allowing f.x. 1000 threads to share only 100 Lua states, thus using only 10% of the memory required by the thread scope.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_macro

<b>Description:</b>	Provides macros within apache httpd runtime configuration files
<b>Status:</b>	Base
<b>Module Identifier:</b>	macro_module
<b>Source File:</b>	mod_macro.c

### Summary

Provides macros within Apache httpd runtime configuration files, to ease the process of creating numerous similar configuration blocks. When the server starts up, the macros are expanded using the provided parameters, and the result is processed as along with the rest of the configuration file.



Macros are defined using `<Macro>` blocks, which contain the portion of your configuration that needs to be repeated, complete with variables for those parts that will need to be substituted.

For example, you might use a macro to define a `<VirtualHost>` block, in order to define multiple similar virtual hosts:

```
<Macro VHost $name $domain>
<VirtualHost *:80>
    ServerName $domain
    ServerAlias www.$domain

    DocumentRoot "/var/www/vhosts/$name"
    ErrorLog "/var/log/httpd/$name.error_log"
    CustomLog "/var/log/httpd/$name.access_log"
</VirtualHost>
</Macro>
```

Macro names are case-insensitive, like httpd configuration directives. However, variable names are case sensitive.

You would then invoke this macro several times to create virtual hosts:

```
Use VHost example example.com
Use VHost myhost hostname.org
Use VHost apache apache.org

UndefMacro VHost
```

At server startup time, each of these `Use` invocations would be expanded into a full virtualhost, as described by the `<Macro>` definition.

The [UndefMacro](#) directive is used so that later macros using the same variable names don't result in conflicting definitions.

A more elaborate version of this example may be seen below in the Examples section.



Parameter names should begin with a sigil such as \$, %, or @, so that they are clearly identifiable, and also in order to help deal with interactions with other directives, such as the core [Define](#) directive. Failure to do so will result in a warning. Nevertheless, you are encouraged to have a good knowledge of your entire server configuration in order to avoid reusing the same variables in different scopes, which can cause confusion.

Parameters prefixed with either \$ or % are not escaped.  
Parameters prefixed with @ are escaped in quotes.

Avoid using a parameter which contains another parameter as a prefix, (For example, \$win and \$winter) as this may cause confusion at expression evaluation time. In the event of such confusion, the longest possible parameter name is used.

If you want to use a value within another string, it is useful to surround the parameter in braces, to avoid confusion:

```
<Macro DocRoot ${docroot}>  
    DocumentRoot "/var/www/${docroot}/htdocs"  
</Macro>
```



## Virtual Host Definition

A common usage of `mod_macro` is for the creation of dynamically-generated virtual hosts.

```
## Define a VHost Macro for repetitive conf:

<Macro VHost $host $port $dir>
  Listen $port
  <VirtualHost *:$port>

    ServerName $host
    DocumentRoot "$dir"

    # Public document root
    <Directory "$dir">
      Require all granted
    </Directory>

    # limit access to intranet subdir.
    <Directory "$dir/intranet">
      Require ip 10.0.0.0/8
    </Directory>
  </VirtualHost>
</Macro>

## Use of VHost with different arguments.

Use VHost www.apache.org 80 /vhosts/apache/
Use VHost example.org 8080 /vhosts/example/
Use VHost www.example.fr 1234 /vhosts/examp
```

## Removal of a macro definition

It's recommended that you undefine a macro once you've used it. This avoids confusion in a complex configuration file where there may be conflicts in variable names.

```
<Macro DirGroup $dir $group>  
  <Directory "$dir">  
    Require group $group  
  </Directory>  
</Macro>
```

```
Use DirGroup /www/apache/private private  
Use DirGroup /www/apache/server admin
```

```
UndefMacro DirGroup
```



<b>Description:</b>	Define a configuration file macro
<b>Syntax:</b>	<code>&lt;Macro <i>name</i> [<i>par1</i> .. <i>parN</i>]&gt; ...</code> <code>&lt;/Macro&gt;</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_macro

The `<Macro>` directive controls the definition of a macro within the server runtime configuration files. The first argument is the name of the macro. Other arguments are parameters to the macro. It is good practice to prefix parameter names with any of '\$%@', and not macro names with such characters.

```
<Macro LocalAccessPolicy>
    Require ip 10.2.16.0/24
</Macro>

<Macro RestrictedAccessPolicy $ipnumbers>
    Require ip $ipnumbers
</Macro>
```



<b>Description:</b>	Undefine a macro
<b>Syntax:</b>	UndefMacro <i>name</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_macro

The **UndefMacro** directive undefines a macro which has been defined before hand.

```
UndefMacro LocalAccessPolicy  
UndefMacro RestrictedAccessPolicy
```



<b>Description:</b>	Use a macro
<b>Syntax:</b>	Use <i>name</i> [ <i>value1</i> ... <i>valueN</i> ]
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_macro

The **Use** directive controls the use of a macro. The specified macro is expanded. It must be given the same number of arguments as in the macro definition. The provided values are associated to their corresponding initial parameters and are substituted before processing.

```
Use LocalAccessPolicy
...
Use RestrictedAccessPolicy "192.54.172.0/24
```

is equivalent, with the macros defined above, to:

```
Require ip 10.2.16.0/24
...
Require ip 192.54.172.0/24 192.54.148.0/24
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_mime`

<b>Description:</b>	Associates the requested filename's extensions with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding)
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>mime_module</code>
<b>Source File:</b>	<code>mod_mime.c</code>

### Summary

This module is used to assign content metadata to the content selected for an HTTP response by mapping patterns in the URI or filenames to the metadata values. For example, the filename extensions of content files often define the content's Internet media type, language, character set, and content-encoding. This information is sent in HTTP messages containing that content and used in content negotiation when selecting alternatives, such that the user's preferences are respected when choosing one of several possible contents to serve. See [mod\\_negotiation](#) for more information about [content negotiation](#).

The directives [AddCharset](#), [AddEncoding](#), [AddLanguage](#) and [AddType](#) are all used to map file extensions onto the metadata for that file. Respectively they set the character set, content-encoding, content-language, and [media-type](#) (content-type) of documents. The directive [TypesConfig](#) is used to specify a file which also maps extensions onto media types.

In addition, [mod\\_mime](#) may define the [handler](#) and [filters](#) that originate and process content. The directives [AddHandler](#), [AddOutputFilter](#), and [AddInputFilter](#) control the modules or

scripts that serve the document. The [MultiviewsMatch](#) directive allows [mod\\_negotiation](#) to consider these file extensions to be included when testing Multiviews matches.

While [mod\\_mime](#) associates metadata with filename extensions, the [core](#) server provides directives that are used to associate all the files in a given container (e.g., [<Location>](#), [<Directory>](#), or [<Files>](#)) with particular metadata. These directives include [ForceType](#), [SetHandler](#), [SetInputFilter](#), and [SetOutputFilter](#). The core directives override any filename extension mappings defined in [mod\\_mime](#).

Note that changing the metadata for a file does not change the value of the Last-Modified header. Thus, previously cached copies may still be used by a client or proxy, with the previous headers. If you change the metadata (language, content type, character set or encoding) you may need to 'touch' affected files (updating their last modified date) to ensure that all visitors are receive the corrected content headers.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[MimeMagicFile](#)

[AddDefaultCharset](#)

ForceType  
SetHandler  
SetInputFilter  
SetOutputFilter



## FILES WITH MULTIPLE EXTENSIONS

Files can have more than one extension; the order of the extensions is *normally* irrelevant. For example, if the file `welcome.html.fr` maps onto content type `text/html` and language French then the file `welcome.fr.html` will map onto exactly the same information. If more than one extension is given that maps onto the same type of metadata, then the one to the right will be used, except for languages and content encodings. For example, if `.gif` maps to the [media-type](#) `image/gif` and `.html` maps to the media-type `text/html`, then the file `welcome.gif.html` will be associated with the media-type `text/html`.

[Languages](#) and [content encodings](#) are treated accumulative, because one can assign more than one language or encoding to a particular resource. For example, the file `welcome.html.en.de` will be delivered with `Content-Language: en, de` and `Content-Type: text/html`.

Care should be taken when a file with multiple extensions gets associated with both a [media-type](#) and a handler. This will usually result in the request being handled by the module associated with the handler. For example, if the `.imap` extension is mapped to the handler `imap-file` (from [mod\\_imagemap](#)) and the `.html` extension is mapped to the media-type `text/html`, then the file `world.imap.html` will be associated with both the `imap-file` handler and `text/html` media-type. When it is processed, the `imap-file` handler will be used, and so it will be treated as a [mod\\_imagemap](#) imagemap file.

If you would prefer only the last dot-separated part of the filename to be mapped to a particular piece of meta-data, then do not use the `Add*` directives. For example, if you wish to have the file `foo.html.cgi` processed as a CGI script, but not the file

bar.cgi.html, then instead of using `AddHandler cgi-script .cgi`, use

### Configure handler based on final extension only

```
<FilesMatch "[^.]+">  
  SetHandler cgi-script  
</FilesMatch>
```



A file of a particular [media-type](#) can additionally be encoded a particular way to simplify transmission over the Internet. While this usually will refer to compression, such as `gzip`, it can also refer to encryption, such as `pgp` or to an encoding such as `UUencoding`, which is designed for transmitting a binary file in an ASCII (text) format.

The [HTTP/1.1 RFC](#), section 14.11 puts it this way:

*The Content-Encoding entity-header field is used as a modifier to the media-type. When present, its value indicates what additional content codings have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a document to be compressed without losing the identity of its underlying media type.*

By using more than one file extension (see [section above about multiple file extensions](#)), you can indicate that a file is of a particular *type*, and also has a particular *encoding*.

For example, you may have a file which is a Microsoft Word document, which is `pkzipped` to reduce its size. If the `.doc` extension is associated with the Microsoft Word file type, and the `.zip` extension is associated with the `pkzip` file encoding, then the file `Resume.doc.zip` would be known to be a `pkzip`'ed Word document.

Apache sends a `Content-encoding` header with the resource, in order to tell the client browser about the encoding method.

```
Content-encoding: pkzip
```



## Character sets and language

In addition to file type and the file encoding, another important piece of information is what language a particular document is in, and in what character set the file should be displayed. For example, the document might be written in the Vietnamese alphabet, or in Cyrillic, and should be displayed as such. This information, also, is transmitted in HTTP headers.

The character set, language, encoding and mime type are all used in the process of content negotiation (See [mod\\_negotiation](#)) to determine which document to give to the client, when there are alternative documents in more than one character set, language, encoding or mime type. All filename extensions associations created with [AddCharset](#), [AddEncoding](#), [AddLanguage](#) and [AddType](#) directives (and extensions listed in the [MimeMagicFile](#)) participate in this select process. Filename extensions that are only associated using the [AddHandler](#), [AddInputFilter](#) or [AddOutputFilter](#) directives may be included or excluded from matching by using the [MultiviewsMatch](#) directive.

### Charset

To convey this further information, Apache optionally sends a Content - Language header, to specify the language that the document is in, and can append additional information onto the Content - Type header to indicate the particular character set that should be used to correctly render the information.

```
Content-Language: en, fr Content-Type: text/plain; charset=ISO-8859-1
```

The language specification is the two-letter abbreviation for the language. The char set is the name of the particular character

set which should be used.



<b>Description:</b>	Maps the given filename extensions to the specified content charset
<b>Syntax:</b>	<code>AddCharset charset extension [extension] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **AddCharset** directive maps the given filename extensions to the specified content charset (the Internet registered name for a given character encoding). *charset* is the [media type's charset parameter](#) for resources with filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

### Example

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

Then the document `xxxx.ja.jis` will be treated as being a Japanese document whose charset is ISO-2022-JP (as will the document `xxxx.jis.ja`). The **AddCharset** directive is useful for both to inform the client about the character encoding of the document so that the document can be interpreted and displayed appropriately, and for [content negotiation](#), where the server returns one from several documents based on the client's charset preference.

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple](#)

[extensions](#) and the *extension* argument will be compared against each of them.

## See also

- [mod\\_negotiation](#)
- [AddDefaultCharset](#)



<b>Description:</b>	Maps the given filename extensions to the specified encoding type
<b>Syntax:</b>	AddEncoding <i>encoding extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **AddEncoding** directive maps the given filename extensions to the specified HTTP content-encoding. *encoding* is the HTTP content coding to append to the value of the Content-Encoding header field for documents named with the *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

**Example**

```
AddEncoding x-gzip .gz
AddEncoding x-compress .Z
```

This will cause filenames containing the .gz extension to be marked as encoded using the x-gzip encoding, and filenames containing the .Z extension to be marked as encoded with x-compress.

Old clients expect x-gzip and x-compress, however the standard dictates that they're equivalent to gzip and compress respectively. Apache does content encoding comparisons by ignoring any leading x-. When responding with an encoding Apache will use whatever form (*i.e.*, x-foo or foo) the client requested. If the client didn't specifically request a particular form Apache will use the form given by the AddEncoding directive. To

make this long story short, you should always use `x-gzip` and `x-compress` for these two specific encodings. More recent encodings, such as `deflate`, should be specified without the `x-`.

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple extensions](#) and the *extension* argument will be compared against each of them.



<b>Description:</b>	Maps the filename extensions to the specified handler
<b>Syntax:</b>	<code>AddHandler <i>handler-name</i> <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

Files having the name *extension* will be served by the specified [handler-name](#). This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*. For example, to activate CGI scripts with the file extension `.cgi`, you might use:

```
AddHandler cgi-script .cgi
```

Once that has been put into your `httpd.conf` file, any file containing the `.cgi` extension will be treated as a CGI program.

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple extensions](#) and the *extension* argument will be compared against each of them.

## See also

- [SetHandler](#)



<b>Description:</b>	Maps filename extensions to the filters that will process client requests
<b>Syntax:</b>	<code>AddInputFilter <i>filter</i> [<i>;filter...</i>] <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

`AddInputFilter` maps the filename extension *extension* to the [filters](#) which will process client requests and POST input when they are received by the server. This is in addition to any filters defined elsewhere, including the `SetInputFilter` directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same *extension*.

If more than one *filter* is specified, they must be separated by semicolons in the order in which they should process the content. The *filter* is case-insensitive.

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple extensions](#) and the *extension* argument will be compared against each of them.

## See also

- [RemoveInputFilter](#)
- [SetInputFilter](#)



## AddLanguage Directive

<b>Description:</b>	Maps the given filename extension to the specified content language
<b>Syntax:</b>	AddLanguage <i>language-tag extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **AddLanguage** directive maps the given filename extension to the specified content language. Files with the filename *extension* are assigned an HTTP Content-Language value of *language-tag* corresponding to the language identifiers defined by RFC 3066. This directive overrides any mappings that already exist for the same *extension*.

### Example

```
AddEncoding x-compress .Z
AddLanguage en .en
AddLanguage fr .fr
```

Then the document `xxxx.en.Z` will be treated as being a compressed English document (as will the document `xxxx.Z.en`). Although the content language is reported to the client, the browser is unlikely to use this information. The **AddLanguage** directive is more useful for [content negotiation](#), where the server returns one from several documents based on the client's language preference.

If multiple language assignments are made for the same extension, the last one encountered is the one that is used. That is, for the case of:

```
AddLanguage en .en  
AddLanguage en-gb .en  
AddLanguage en-us .en
```

documents with the extension `.en` would be treated as being `en-us`.

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple extensions](#) and the *extension* argument will be compared against each of them.

## See also

- [mod\\_negotiation](#)



## AddOutputFilter Directive

<b>Description:</b>	Maps filename extensions to the filters that will process responses from the server
<b>Syntax:</b>	<code>AddOutputFilter <i>filter</i> [<i>;filter...</i>] <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The `AddOutputFilter` directive maps the filename extension *extension* to the [filters](#) which will process responses from the server before they are sent to the client. This is in addition to any filters defined elsewhere, including `SetOutputFilter` and `AddOutputFilterByType` directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same *extension*.

For example, the following configuration will process all `.html` files for server-side includes and will then compress the output using `mod_deflate`.

```
AddOutputFilter INCLUDES;DEFLATE html
```

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content. The *filter* argument is case-insensitive.

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple extensions](#) and the *extension* argument will be compared against each of them.

Note that when defining a set of filters using the

[AddOutputFilter](#) directive, any definition made will replace any previous definition made by the [AddOutputFilter](#) directive.

```
# Effective filter "DEFLATE"
AddOutputFilter DEFLATE shtml
<Location "/foo">
  # Effective filter "INCLUDES", replacing '
  AddOutputFilter INCLUDES shtml
</Location>
<Location "/bar">
  # Effective filter "INCLUDES;DEFLATE", re
  AddOutputFilter INCLUDES;DEFLATE shtml
</Location>
<Location "/bar/baz">
  # Effective filter "BUFFER", replacing "I
  AddOutputFilter BUFFER shtml
</Location>
<Location "/bar/baz/buz">
  # No effective filter, replacing "BUFFER"
  RemoveOutputFilter shtml
</Location>
```

## See also

- [RemoveOutputFilter](#)
- [SetOutputFilter](#)



## AddType Directive

**Description:** Maps the given filename extensions onto the specified content type

**Syntax:** `AddType media-type extension [extension] ...`

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Status:** Base

**Module:** mod\_mime

The **AddType** directive maps the given filename extensions onto the specified content type. *media-type* is the [media type](#) to use for filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

It is recommended that new media types be added using the **AddType** directive rather than changing the [TypesConfig](#) file.

### Example

```
AddType image/gif .gif
```

Or, to specify multiple file extensions in one directive:

### Example

```
AddType image/jpeg jpeg jpg jpe
```

The *extension* argument is case-insensitive and can be specified with or without a leading dot. Filenames may have [multiple extensions](#) and the *extension* argument will be compared against each of them.

A similar effect to [mod\\_negotiation's LanguagePriority](#) can be achieved by qualifying a *media-type* with `qs`:

### Example

```
AddType application/rss+xml;qs=0.8 .xml
```

This is useful in situations, e.g. when a client requesting `Accept: */*` can not actually process the content returned by the server.

This directive primarily configures the content types generated for static files served out of the filesystem. For resources other than static files, where the generator of the response typically specifies a Content-Type, this directive has no effect.

### Note

If no handler is explicitly set for a request, the specified content type will also be used as the handler name.

When explicit directives such as [SetHandler](#) or [AddHandler](#) do not apply to the current request, the internal handler name normally set by those directives is instead set to the content type specified by this directive.

This is a historical behavior that may be used by some third-party modules (such as `mod_php`) for taking responsibility for the matching request.

Configurations that rely on such "synthetic" types should be avoided. Additionally, configurations that restrict access to [SetHandler](#) or [AddHandler](#) should restrict access to this directive as well.

### See also

- [ForceType](#)

- mod\_negotiation



## Content-Language Directive

<b>Description:</b>	Defines a default language-tag to be sent in the Content-Language header field for all resources in the current context that have not been assigned a language-tag by some other means.
<b>Syntax:</b>	DefaultLanguage <i>language-tag</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The `DefaultLanguage` directive tells Apache that all resources in the directive's scope (e.g., all resources covered by the current `<Directory>` container) that don't have an explicit language extension (such as `.fr` or `.de` as configured by `AddLanguage`) should be assigned a Content-Language of *language-tag*. This allows entire directory trees to be marked as containing Dutch content, for instance, without having to rename each file. Note that unlike using extensions to specify languages, `DefaultLanguage` can only specify a single language.

If no `DefaultLanguage` directive is in force and a file does not have any language extensions as configured by `AddLanguage`, then no Content-Language header field will be generated.

### Example

```
DefaultLanguage en
```

### See also

- [mod\\_negotiation](#)



<b>Description:</b>	Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename
<b>Syntax:</b>	<code>ModMimeUsePathInfo On Off</code>
<b>Default:</b>	<code>ModMimeUsePathInfo Off</code>
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_mime</code>

The `ModMimeUsePathInfo` directive is used to combine the filename with the `path_info` URL component to apply `mod_mime`'s directives to the request. The default value is `Off` - therefore, the `path_info` component is ignored.

This directive is recommended when you have a virtual filesystem.

### Example

```
ModMimeUsePathInfo On
```

If you have a request for `/index.php/foo.shtml` `mod_mime` will now treat the incoming request as `/index.php/foo.shtml` and directives like `AddOutputFilter INCLUDES .shtml` will add the `INCLUDES` filter to the request. If `ModMimeUsePathInfo` is not set, the `INCLUDES` filter will not be added. This will work analogously for virtual paths, such as those defined by `<Location>`

### See also

- [AcceptPathInfo](#)



<b>Description:</b>	The types of files that will be included when searching for a matching file with MultiViews
<b>Syntax:</b>	MultiViewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters]
<b>Default:</b>	MultiViewsMatch NegotiatedOnly
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

`MultiViewsMatch` permits three different behaviors for `mod_negotiation`'s MultiViews feature. MultiViews allows a request for a file, e.g. `index.html`, to match any negotiated extensions following the base request, e.g. `index.html.en`, `index.html.fr`, or `index.html.gz`.

The `NegotiatedOnly` option provides that every extension following the base name must correlate to a recognized `mod_mime` extension for content negotiation, e.g. `Charset`, `Content-Type`, `Language`, or `Encoding`. This is the strictest implementation with the fewest unexpected side effects, and is the default behavior.

To include extensions associated with `Handlers` and/or `Filters`, set the `MultiViewsMatch` directive to either `Handlers`, `Filters`, or both option keywords. If all other factors are equal, the smallest file will be served, e.g. in deciding between `index.html.cgi` of 500 bytes and `index.html.pl` of 1000 bytes, the `.cgi` file would win in this example. Users of `.asis` files might prefer to use the `Handler` option, if `.asis` files are associated with the `asis-handler`.

You may finally allow Any extensions to match, even if [mod\\_mime](#) doesn't recognize the extension. This can cause unpredictable results, such as serving .old or .bak files the webmaster never expected to be served.

For example, the following configuration will allow handlers and filters to participate in Multiviews, but will exclude unknown files:

```
MultiviewsMatch Handlers Filters
```

`MultiviewsMatch` is not allowed in a [<Location>](#) or [<LocationMatch>](#) section.

## See also

- [Options](#)
- [mod\\_negotiation](#)



<b>Description:</b>	Removes any character set associations for a set of file extensions
<b>Syntax:</b>	RemoveCharset <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveCharset** directive removes any character set associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive and can be specified with or without a leading dot.

### Example

```
RemoveCharset .html .shtml
```



<b>Description:</b>	Removes any content encoding associations for a set of file extensions
<b>Syntax:</b>	RemoveEncoding <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveEncoding** directive removes any encoding associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

**/foo/.htaccess:**

```
AddEncoding x-gzip .gz
AddType text/plain .asc
<Files "*.gz.asc">
    RemoveEncoding .gz
</Files>
```

This will cause foo.gz to be marked as being encoded with the gzip method, but foo.gz.asc as an unencoded plaintext file.

**Note**

**RemoveEncoding** directives are processed *after* any **AddEncoding** directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

The *extension* argument is case-insensitive and can be specified

with or without a leading dot.



<b>Description:</b>	Removes any handler associations for a set of file extensions
<b>Syntax:</b>	RemoveHandler <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveHandler** directive removes any handler associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

**/foo/.htaccess:**

```
AddHandler server-parsed .html
```

**/foo/bar/.htaccess:**

```
RemoveHandler .html
```

This has the effect of returning .html files in the /foo/bar directory to being treated as normal files, rather than as candidates for parsing (see the [mod\\_include](#) module).

The *extension* argument is case-insensitive and can be specified with or without a leading dot.



<b>Description:</b>	Removes any input filter associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveInputFilter <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The `RemoveInputFilter` directive removes any input [filter](#) associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive and can be specified with or without a leading dot.

## See also

- [AddInputFilter](#)
- [SetInputFilter](#)



## RemoveLanguage Directive

<b>Description:</b>	Removes any language associations for a set of file extensions
<b>Syntax:</b>	RemoveLanguage <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveLanguage** directive removes any language associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive and can be specified with or without a leading dot.



## RemoveOutputFilter Directive

<b>Description:</b>	Removes any output filter associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveOutputFilter <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The `RemoveOutputFilter` directive removes any output [filter](#) associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive and can be specified with or without a leading dot.

### Example

```
RemoveOutputFilter shtml
```

### See also

- [AddOutputFilter](#)



<b>Description:</b>	Removes any content type associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveType <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveType** directive removes any [media type](#) associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

**/foo/.htaccess:**

```
RemoveType .cgi
```

This will remove any special handling of .cgi files in the /foo/ directory and any beneath it, causing responses containing those files to omit the HTTP Content-Type header field.

**Note**

**RemoveType** directives are processed *after* any [AddType](#) directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

The *extension* argument is case-insensitive and can be specified with or without a leading dot.



## TypesConfig Directive

<b>Description:</b>	The location of the <code>mime.types</code> file
<b>Syntax:</b>	<code>TypesConfig file-path</code>
<b>Default:</b>	<code>TypesConfig conf/mime.types</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_mime</code>

The `TypesConfig` directive sets the location of the [media types](#) configuration file. *File-path* is relative to the [ServerRoot](#). This file sets the default list of mappings from filename extensions to content types. Most administrators use the `mime.types` file provided by their OS, which associates common filename extensions with the official list of IANA registered media types maintained at <http://www.iana.org/assignments/media-types/index.html> as well as a large number of unofficial types. This simplifies the `httpd.conf` file by providing the majority of media-type definitions, and may be overridden by `AddType` directives as needed. You should not edit the `mime.types` file, because it may be replaced when you upgrade your server.

The file contains lines in the format of the arguments to an `AddType` directive:

```
media-type [extension] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (`#`) are ignored. Empty lines are there for completeness (of the `mime.types` file). Apache `httpd` can still determine these types with [mod\\_mime\\_magic](#).

Please do **not** send requests to the Apache HTTP Server Project to add any new entries in the distributed `mime.types`

file unless (1) they are already registered with IANA, and (2) they use widely accepted, non-conflicting filename extensions across platforms. category/x-subtype requests will be automatically rejected, as will any new two-letter extensions as they will likely conflict later with the already crowded language and character set namespace.

## See also

- [mod\\_mime\\_magic](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_mime_magic`

<b>Description:</b>	Determines the MIME type of a file by looking at a few bytes of its contents
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>mime_magic_module</code>
<b>Source File:</b>	<code>mod_mime_magic.c</code>

### Summary

This module determines the [MIME type](#) of files in the same way the Unix `file(1)` command works: it looks at the first few bytes of the file. It is intended as a "second line of defense" for cases that [mod\\_mime](#) can't resolve.

This module is derived from a free version of the `file(1)` command for Unix, which uses "magic numbers" and other hints from a file's contents to figure out what the contents are. This module is active only if the magic file is specified by the [MimeMagicFile](#) directive.



The contents of the file are plain ASCII text in 4-5 columns. Blank lines are allowed but ignored. Commented lines use a hash mark (#). The remaining lines are parsed for the following columns:

Column	Description																						
1	byte number to begin checking from ">" indicates a dependency upon the previous non->" line																						
2	type of data to match <table border="1"> <tbody> <tr> <td>byte</td> <td>single character</td> </tr> <tr> <td>short</td> <td>machine-order 16-bit integer</td> </tr> <tr> <td>long</td> <td>machine-order 32-bit integer</td> </tr> <tr> <td>string</td> <td>arbitrary-length string</td> </tr> <tr> <td>date</td> <td>long integer date (seconds since Unix epoch/1970)</td> </tr> <tr> <td>beshort</td> <td>big-endian 16-bit integer</td> </tr> <tr> <td>belong</td> <td>big-endian 32-bit integer</td> </tr> <tr> <td>bedate</td> <td>big-endian 32-bit integer date</td> </tr> <tr> <td>leshort</td> <td>little-endian 16-bit integer</td> </tr> <tr> <td>lelong</td> <td>little-endian 32-bit integer</td> </tr> <tr> <td>ledate</td> <td>little-endian 32-bit integer date</td> </tr> </tbody> </table>	byte	single character	short	machine-order 16-bit integer	long	machine-order 32-bit integer	string	arbitrary-length string	date	long integer date (seconds since Unix epoch/1970)	beshort	big-endian 16-bit integer	belong	big-endian 32-bit integer	bedate	big-endian 32-bit integer date	leshort	little-endian 16-bit integer	lelong	little-endian 32-bit integer	ledate	little-endian 32-bit integer date
byte	single character																						
short	machine-order 16-bit integer																						
long	machine-order 32-bit integer																						
string	arbitrary-length string																						
date	long integer date (seconds since Unix epoch/1970)																						
beshort	big-endian 16-bit integer																						
belong	big-endian 32-bit integer																						
bedate	big-endian 32-bit integer date																						
leshort	little-endian 16-bit integer																						
lelong	little-endian 32-bit integer																						
ledate	little-endian 32-bit integer date																						
3	contents of data to match																						
4	MIME type if matched																						
5	MIME encoding if matched (optional)																						

For example, the following magic file lines would recognize some audio formats:

```
# Sun/NeXT audio data
```

```

0      string      .snd
>12   belong      1      audio/basic
>12   belong      2      audio/basic
>12   belong      3      audio/basic
>12   belong      4      audio/basic
>12   belong      5      audio/basic
>12   belong      6      audio/basic
>12   belong      7      audio/basic
>12   belong      23     audio/x-adpcm

```

Or these would recognize the difference between \* .doc files containing Microsoft Word or FrameMaker documents. (These are incompatible file formats which use the same file suffix.)

```

# Frame
0  string  \<MakerFile      application/x-frame
0  string  \<MIFFile      application/x-frame
0  string  \<MakerDictionary  application/x-frame
0  string  \<MakerScreenFon  application/x-frame
0  string  \<MML      application/x-frame
0  string  \<Book      application/x-frame
0  string  \<Maker      application/x-frame

# MS-Word
0  string  \376\067\0\043      application/msword
0  string  \320\317\021\340\241\261  application/msword
0  string  \333\245-\0\0\0      application/msword

```

An optional MIME encoding can be included as a fifth column. For example, this can recognize gzipped files and set the encoding for them.

```

# gzip (GNU zip, not to be confused with
#      [Info-ZIP/PKWARE] zip archiver)

0  string  \037\213  application/octet-stream  x-gzip

```



This module is not for every system. If your system is barely keeping up with its load or if you're performing a web server benchmark, you may not want to enable this because the processing is not free.

However, an effort was made to improve the performance of the original `file(1)` code to make it fit in a busy web server. It was designed for a server where there are thousands of users who publish their own documents. This is probably very common on intranets. Many times, it's helpful if the server can make more intelligent decisions about a file's contents than the file name allows ...even if just to reduce the "why doesn't my page work" calls when users improperly name their own files. You have to decide if the extra work suits your environment.



The following notes apply to the `mod_mime_magic` module and are included here for compliance with contributors' copyright restrictions that require their acknowledgment.

`mod_mime_magic`: MIME type lookup via file magic numbers  
Copyright (c) 1996-1997 Cisco Systems, Inc.

This software was submitted by Cisco Systems to the Apache Group in July 1997. Future revisions and derivatives of this source code must acknowledge Cisco Systems as the original contributor of this module. All other licensing and usage conditions are those of the Apache Group.

Some of this code is derived from the free version of the file command originally posted to comp.sources.unix. Copyright info for that program is included below as required.

- Copyright (c) Ian F. Darwin, 1987. Written by Ian F. Darwin.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must

not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

For compliance with Mr Darwin's terms: this has been very significantly modified from the free "file" command.

- all-in-one file for compilation convenience when moving from one version of Apache to the next.
- Memory allocation is done through the Apache API's pool structure.
- All functions have had necessary Apache API request or server structures passed to them where necessary to call other Apache API routines. (*i.e.*, usually for logging, files, or memory allocation in itself or a called function.)
- struct magic has been converted from an array to a single-ended linked list because it only grows one record at a time, it's only accessed sequentially, and the Apache API has no equivalent of `realloc()`.
- Functions have been changed to get their parameters from the server configuration instead of globals. (It should be reentrant now but has not been tested in a threaded environment.)
- Places where it used to print results to stdout now saves them in a list where they're used to set the MIME type in the Apache request record.
- Command-line flags have been removed since they will never be used here.



<b>Description:</b>	Enable MIME-type determination based on file contents using the specified magic file
<b>Syntax:</b>	MimeMagicFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_mime_magic

The `MimeMagicFile` directive can be used to enable this module, the default file is distributed at `conf/magic`. Non-rooted paths are relative to the `ServerRoot`. Virtual hosts will use the same file as the main server unless a more specific setting is used, in which case the more specific setting overrides the main server's file.

**Example**

```
MimeMagicFile conf/magic
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_negotiation

<b>Description:</b>	Provides for <a href="#">content negotiation</a>
<b>Status:</b>	Base
<b>Module Identifier:</b>	negotiation_module
<b>Source File:</b>	mod_negotiation.c

## Summary

Content negotiation, or more accurately content selection, is the selection of the document that best matches the clients capabilities, from one of several available documents. There are two implementations of this.

- A type map (a file with the handler type-map) which explicitly lists the files containing the variants.
- A Multiviews search (enabled by the Multiviews [Options](#)), where the server does an implicit filename pattern match, and choose from amongst the results.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[Options](#)

[mod\\_mime](#)

[Content Negotiation](#)  
[Environment Variables](#)



## Type Maps

A type map has a format similar to RFC822 mail headers. It contains document descriptions separated by blank lines, with lines beginning with a hash character ('#') treated as comments. A document description consists of several header records; records may be continued on multiple lines if the continuation lines start with spaces. The leading space will be deleted and the lines concatenated. A header record consists of a keyword name, which always ends in a colon, followed by a value. Whitespace is allowed between the header name and value, and between the tokens of value. The headers allowed are:

### **Content - Encoding:**

The encoding of the file. Apache only recognizes encodings that are defined by an [AddEncoding](#) directive. This normally includes the encodings x-compress for compress'd files, and x-gzip for gzip'd files. The x- prefix is ignored for encoding comparisons.

### **Content - Language:**

The language(s) of the variant, as an Internet standard language tag ([RFC 1766](#)). An example is en, meaning English. If the variant contains more than one language, they are separated by a comma.

### **Content - Length:**

The length of the file, in bytes. If this header is not present, then the actual length of the file is used.

### **Content - Type:**

The [MIME media type](#) of the document, with optional parameters. Parameters are separated from the media type and from one another by a semi-colon, with a syntax of name=value. Common parameters include:

#### **level**

an integer specifying the version of the media type. For

text/html this defaults to 2, otherwise 0.

## **qs**

a floating-point number with a value in the range 0[.000] to 1[.000], indicating the relative 'quality' of this variant compared to the other available variants, independent of the client's capabilities. For example, a jpeg file is usually of higher source quality than an ascii file if it is attempting to represent a photograph. However, if the resource being represented is ascii art, then an ascii file would have a higher source quality than a jpeg file. All qs values are therefore specific to a given resource.

### **Example**

```
Content-Type: image/jpeg; qs=0.8
```

## **URI :**

uri of the file containing the variant (of the given media type, encoded with the given content encoding). These are interpreted as URLs relative to the map file; they must be on the same server, and they must refer to files to which the client would be granted access if they were to be requested directly.

## **Body :**

The actual content of the resource may be included in the type-map file using the Body header. This header must contain a string that designates a delimiter for the body content. Then all following lines in the type map file will be considered part of the resource body until the delimiter string is found.

### **Example:**

```
Body: ----xyz----  
<html>
```

```
<body>
<p>Content of the page.</p>
</body>
</html>
----xyz----
```

Consider, for example, a resource called `document.html` which is available in English, French, and German. The files for each of these are called `document.html.en`, `document.html.fr`, and `document.html.de`, respectively. The type map file will be called `document.html.var`, and will contain the following:

```
URI: document.html

Content-language: en
Content-type: text/html
URI: document.html.en

Content-language: fr
Content-type: text/html
URI: document.html.fr

Content-language: de
Content-type: text/html
URI: document.html.de
```

All four of these files should be placed in the same directory, and the `.var` file should be associated with the `type-map` handler with an [AddHandler](#) directive:

```
AddHandler type-map .var
```

A request for `document.html.var` in this directory will result in choosing the variant which most closely matches the language preference specified in the user's `Accept-Language` request header.

If `Multiviews` is enabled, and [MultiviewsMatch](#) is set to

"handlers" or "any", a request to document .html will discover document .html.var and continue negotiating with the explicit type map.

Other configuration directives, such as [Alias](#) can be used to map document .html to document .html.var.



---

A Multiviews search is enabled by the Multiviews [Options](#). If the server receives a request for `/some/dir/foo` and `/some/dir/foo` does *not* exist, then the server reads the directory looking for all files named `foo.*`, and effectively fakes up a type map which names all those files, assigning them the same media types and content-encodings it would have if the client had asked for one of them by name. It then chooses the best match to the client's requirements, and returns that document.

The [MultiviewsMatch](#) directive configures whether Apache will consider files that do not have content negotiation meta-information assigned to them when choosing files.



## CacheNegotiatedDocs Directive

<b>Description:</b>	Allows content-negotiated documents to be cached by proxy servers
<b>Syntax:</b>	CacheNegotiatedDocs On Off
<b>Default:</b>	CacheNegotiatedDocs Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_negotiation

If set, this directive allows content-negotiated documents to be cached by proxy servers. This could mean that clients behind those proxys could retrieve versions of the documents that are not the best match for their abilities, but it will make caching more efficient.

This directive only applies to requests which come from HTTP/1.0 browsers. HTTP/1.1 provides much better control over the caching of negotiated documents, and this directive has no effect in responses to HTTP/1.1 requests.



## ForceLanguagePriority Directive

<b>Description:</b>	Action to take if a single acceptable document is not found
<b>Syntax:</b>	ForceLanguagePriority None   Prefer   Fallback [Prefer   Fallback]
<b>Default:</b>	ForceLanguagePriority Prefer
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_negotiation

The `ForceLanguagePriority` directive uses the given `LanguagePriority` to satisfy negotiation where the server could otherwise not return a single matching document.

`ForceLanguagePriority Prefer` uses `LanguagePriority` to serve a one valid result, rather than returning an HTTP result 300 (MULTIPLE CHOICES) when there are several equally valid choices. If the directives below were given, and the user's `Accept-Language` header assigned `en` and `de` each as quality `.500` (equally acceptable) then the first matching variant, `en`, will be served.

```
LanguagePriority en fr de
ForceLanguagePriority Prefer
```

`ForceLanguagePriority Fallback` uses `LanguagePriority` to serve a valid result, rather than returning an HTTP result 406 (NOT ACCEPTABLE). If the directives below were given, and the user's `Accept-Language` only permitted an `es` language response, but such a variant isn't found, then the first variant from the `LanguagePriority` list below will be served.

LanguagePriority en fr de  
ForceLanguagePriority Fallback

Both options, `Prefer` and `Fallback`, may be specified, so either the first matching variant from [LanguagePriority](#) will be served if more than one variant is acceptable, or first available document will be served if none of the variants matched the client's acceptable list of languages.

## See also

- [AddLanguage](#)



<b>Description:</b>	The precedence of language variants for cases where the client does not express a preference
<b>Syntax:</b>	LanguagePriority <i>MIME-lang</i> [ <i>MIME-lang</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_negotiation

The `LanguagePriority` sets the precedence of language variants for the case where the client does not express a preference, when handling a Multiviews request. The list of *MIME-lang* are in order of decreasing preference.

```
LanguagePriority en fr de
```

For a request for `foo.html`, where `foo.html.fr` and `foo.html.de` both existed, but the browser did not express a language preference, then `foo.html.fr` would be returned.

Note that this directive only has an effect if a 'best' language cannot be determined by any other means or the `ForceLanguagePriority` directive is not `None`. In general, the client determines the language preference, not the server.

## See also

- [AddLanguage](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_nw\_ssl

<b>Description:</b>	Enable SSL encryption for NetWare
<b>Status:</b>	Base
<b>Module Identifier:</b>	nwssl_module
<b>Source File:</b>	mod_nw_ssl.c
<b>Compatibility:</b>	NetWare only

## Summary

This module enables SSL encryption for a specified port. It takes advantage of the SSL encryption functionality that is built into the NetWare operating system.



<b>Description:</b>	List of additional client certificates
<b>Syntax:</b>	NWSSLTrustedCerts <i>filename</i> [ <i>filename</i> ] ...
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Specifies a list of client certificate files (DER format) that are used when creating a proxied SSL connection. Each client certificate used by a server must be listed separately in its own .der file.



<b>Description:</b>	Allows a connection to be upgraded to an SSL connection upon request
<b>Syntax:</b>	NWSSLUpgradeable [ <i>IP-address:</i> ]portnumber
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Allow a connection that was created on the specified address and/or port to be upgraded to an SSL connection upon request from the client. The address and/or port must have already be defined previously with a [Listen](#) directive.



<b>Description:</b>	Enables SSL encryption for the specified port
<b>Syntax:</b>	<code>SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Specifies the port and the eDirectory based certificate name that will be used to enable SSL encryption. An optional third parameter also enables mutual authentication.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_privileges`

<b>Description:</b>	Support for Solaris privileges and for running virtual hosts under different user IDs.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>privileges_module</code>
<b>Source File:</b>	<code>mod_privileges.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and up, on Solaris 10 and OpenSolaris platforms

### Summary

This module enables different Virtual Hosts to run with different Unix *User* and *Group* IDs, and with different [Solaris Privileges](#). In particular, it offers a solution to the problem of privilege separation between different Virtual Hosts, first promised by the abandoned `perchild` MPM. It also offers other security enhancements.

Unlike `perchild`, `mod_privileges` is not itself an MPM. It works *within* a processing model to set privileges and User/Group *per request* in a running process. It is therefore not compatible with a threaded MPM, and will refuse to run under one.

`mod_privileges` raises security issues similar to those of [suexec](#). But unlike `suexec`, it applies not only to CGI programs but to the entire request processing cycle, including in-process applications and subprocesses. It is ideally suited to running PHP applications under `mod_php`, which is also incompatible with threaded MPMs. It is also well-suited to other in-process scripting applications such as `mod_perl`, `mod_python`, and `mod_ruby`, and to applications implemented in C as apache modules where privilege separation is an issue.



## Security Concerns

`mod_privileges` introduces new security concerns in situations where **untrusted code** may be run **within the webserver process**. This applies to untrusted modules, and scripts running under modules such as `mod_php` or `mod_perl`. Scripts running externally (e.g. as CGI or in an appserver behind `mod_proxy` or `mod_jk`) are NOT affected.

The basic security concerns with `mod_privileges` are:

- Running as a system user introduces the same security issues as `mod_suexec`, and near-equivalents such as `cgiwrap` and `suphp`.
- A privileges-aware malicious user extension (module or script) could escalate its privileges to anything available to the `httpd` process in any virtual host. This introduces new risks if (and only if) `mod_privileges` is compiled with the `BIG_SECURITY_HOLE` option.
- A privileges-aware malicious user extension (module or script) could escalate privileges to set its user ID to another system user (and/or group).

The `PrivilegesMode` directive allows you to select either `FAST` or `SECURE` mode. You can mix modes, using `FAST` mode for trusted users and fully-audited code paths, while imposing `SECURE` mode where an untrusted user has scope to introduce code.

Before describing the modes, we should also introduce the target use cases: Benign vs Hostile. In a benign situation, you want to separate users for their convenience, and protect them and the server against the risks posed by honest mistakes, but you trust your users are not deliberately subverting system security. In a hostile situation - e.g. commercial hosting - you may have users deliberately attacking the system or each other.

## **FAST mode**

In *FAST* mode, requests are run in-process with the selected uid/gid and privileges, so the overhead is negligible. This is suitable for benign situations, but is not secure against an attacker escalating privileges with an in-process module or script.

## **SECURE mode**

A request in *SECURE* mode forks a subprocess, which then drops privileges. This is a very similar case to running CGI with suexec, but for the entire request cycle, and with the benefit of fine-grained control of privileges.

You can select different **PrivilegesModes** for each virtual host, and even in a directory context within a virtual host. *FAST* mode is appropriate where the user(s) are trusted and/or have no privilege to load in-process code. *SECURE* mode is appropriate to cases where untrusted code might be run in-process. However, even in *SECURE* mode, there is no protection against a malicious user who is able to introduce privileges-aware code running *before the start of the request-processing cycle*.



## DTracePrivileges Directive

<b>Description:</b>	Determines whether the privileges required by dtrace are enabled.
<b>Syntax:</b>	DTracePrivileges On Off
<b>Default:</b>	DTracePrivileges Off
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM).

This server-wide directive determines whether Apache will run with the [privileges](#) required to run [dtrace](#). Note that *DTracePrivileges On* will not in itself activate DTrace, but *DTracePrivileges Off* will prevent it working.



<b>Description:</b>	Trade off processing speed and efficiency vs security against malicious privileges-aware code.
<b>Syntax:</b>	PrivilegesMode FAST SECURE SELECTIVE
<b>Default:</b>	PrivilegesMode FAST
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM).

This directive trades off performance vs security against malicious, privileges-aware code. In *SECURE* mode, each request runs in a secure subprocess, incurring a substantial performance penalty. In *FAST* mode, the server is not protected against escalation of privileges as discussed above.

This directive differs slightly between a `<Directory>` context (including equivalents such as `Location/Files/If`) and a top-level or `<VirtualHost>`.

At top-level, it sets a default that will be inherited by virtualhosts. In a virtual host, *FAST* or *SECURE* mode acts on the entire HTTP request, and any settings in a `<Directory>` context will be **ignored**. A third pseudo-mode *SELECTIVE* defers the choice of *FAST* vs *SECURE* to directives in a `<Directory>` context.

In a `<Directory>` context, it is applicable only where *SELECTIVE* mode was set for the `VirtualHost`. Only *FAST* or *SECURE* can be set in this context (*SELECTIVE* would be meaningless).

### Warning

Where SELECTIVE mode is selected for a virtual host, the activation of privileges must be deferred until *after* the mapping phase of request processing has determined what <Directory> context applies to the request. This might give an attacker opportunities to introduce code through a [RewriteMap](#) running at top-level or <VirtualHost> context *before* privileges have been dropped and userid/gid set.



<b>Description:</b>	Determines whether the virtualhost can run subprocesses, and the privileges available to subprocesses.
<b>Syntax:</b>	VHostCGIMode On Off Secure
<b>Default:</b>	VHostCGIMode On
<b>Context:</b>	virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM).

Determines whether the virtual host is allowed to run fork and exec, the [privileges](#) required to run subprocesses. If this is set to *Off* the virtualhost is denied the privileges and will not be able to run traditional CGI programs or scripts under the traditional [mod\\_cgi](#), nor similar external programs such as those created by [mod\\_ext\\_filter](#) or [RewriteMap prog](#). Note that it does not prevent CGI programs running under alternative process and security models such as [mod\\_fcgid](#), which is a recommended solution in Solaris.

If set to *On* or *Secure*, the virtual host is permitted to run external programs and scripts as above. Setting [VHostCGIMode Secure](#) has the effect of denying privileges to the subprocesses, as described for [VHostSecure](#).



<b>Description:</b>	Assign arbitrary privileges to subprocesses created by a virtual host.
<b>Syntax:</b>	VHostPrivs [+ -]? <i>privilege-name</i> [[+ -]? <i>privilege-name</i> ] ...
<b>Default:</b>	None
<b>Context:</b>	virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM) and when <a href="#">mod_privileges</a> is compiled with the <i>BIG_SECURITY_HOLE</i> compile-time option.

**VHostCGIPrivs** can be used to assign arbitrary [privileges](#) to subprocesses created by a virtual host, as discussed under **VHostCGIMode**. Each *privilege-name* is the name of a Solaris privilege, such as *file\_setid* or *sys\_nfs*.

A *privilege-name* may optionally be prefixed by + or -, which will respectively allow or deny a privilege. If used with neither + nor -, all privileges otherwise assigned to the virtualhost will be denied. You can use this to override any of the default sets and construct your own privilege set.

**Security**

This directive can open huge security holes in apache subprocesses, up to and including running them with root-level powers. Do not use it unless you fully understand what you are doing!



<b>Description:</b>	Sets the Group ID under which a virtual host runs.
<b>Syntax:</b>	VHostGroup <i>unix-groupid</i>
<b>Default:</b>	Inherits the group id specified in <a href="#">Group</a>
<b>Context:</b>	virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM).

The `VHostGroup` directive sets the Unix group under which the server will process requests to a virtualhost. The group is set before the request is processed and reset afterwards using [Solaris Privileges](#). Since the setting applies to the *process*, this is not compatible with threaded MPMs.

*Unix-group* is one of:

### A group name

Refers to the given group by name.

### # followed by a group number.

Refers to a group by its number.

### Security

This directive cannot be used to run apache as root! Nevertheless, it opens potential security issues similar to those discussed in the [suexec](#) documentation.

### See also

- [Group](#)

- [SuexecUserGroup](#)



<b>Description:</b>	Assign arbitrary privileges to a virtual host.
<b>Syntax:</b>	VHostPrivs [+ -]? <i>privilege-name</i> [[+ -]? <i>privilege-name</i> ] ...
<b>Default:</b>	None
<b>Context:</b>	virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM) and when <a href="#">mod_privileges</a> is compiled with the <i>BIG_SECURITY_HOLE</i> compile-time option.

**VHostPrivs** can be used to assign arbitrary [privileges](#) to a virtual host. Each *privilege-name* is the name of a Solaris privilege, such as *file\_setid* or *sys\_nfs*.

A *privilege-name* may optionally be prefixed by + or -, which will respectively allow or deny a privilege. If used with neither + nor -, all privileges otherwise assigned to the virtualhost will be denied. You can use this to override any of the default sets and construct your own privilege set.

### Security

This directive can open huge security holes in apache, up to and including running requests with root-level powers. Do not use it unless you fully understand what you are doing!



<b>Description:</b>	Determines whether the server runs with enhanced security for the virtualhost.
<b>Syntax:</b>	VHostSecure On Off
<b>Default:</b>	VHostSecure On
<b>Context:</b>	virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM).

Determines whether the virtual host processes requests with security enhanced by removal of [Privileges](#) that are rarely needed in a webserver, but which are available by default to a normal Unix user and may therefore be required by modules and applications. It is recommended that you retain the default (On) unless it prevents an application running. Since the setting applies to the *process*, this is not compatible with threaded MPMs.

#### Note

If **VHostSecure** prevents an application running, this may be a warning sign that the application should be reviewed for security.



<b>Description:</b>	Sets the User ID under which a virtual host runs.
<b>Syntax:</b>	VHostUser <i>unix-userid</i>
<b>Default:</b>	Inherits the userid specified in <a href="#">User</a>
<b>Context:</b>	virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_privileges
<b>Compatibility:</b>	Available on Solaris 10 and OpenSolaris with non-threaded MPMs ( <a href="#">prefork</a> or custom MPM).

The `VHostUser` directive sets the Unix userid under which the server will process requests to a virtualhost. The userid is set before the request is processed and reset afterwards using [Solaris Privileges](#). Since the setting applies to the *process*, this is not compatible with threaded MPMs.

*Unix-userid* is one of:

### A username

Refers to the given user by name.

### # followed by a user number.

Refers to a user by its number.

### Security

This directive cannot be used to run apache as root! Nevertheless, it opens potential security issues similar to those discussed in the [suexec](#) documentation.

### See also

- [User](#)
- [SuexecUserGroup](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy`

<b>Description:</b>	Multi-protocol proxy/gateway server
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_module</code>
<b>Source File:</b>	<code>mod_proxy.c</code>

## Summary

### Warning

Do not enable proxying with [ProxyRequests](#) until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

`mod_proxy` and related modules implement a proxy/gateway for Apache HTTP Server, supporting a number of popular protocols as well as several different load balancing algorithms. Third-party modules can add support for additional protocols and load balancing algorithms.

A set of modules must be loaded into the server to provide the necessary features. These modules can be included statically at build time or dynamically via the [LoadModule](#) directive). The set must include:

- `mod_proxy`, which provides basic proxy capabilities
- `mod_proxy_balancer` and one or more balancer modules if load balancing is required. (See [mod\\_proxy\\_balancer](#) for more information.)
- one or more proxy scheme, or protocol, modules:

Protocol	Module
AJP13 (Apache JServe Protocol)	<code>mod_proxy_ajp</code>

version 1.3)	
CONNECT (for SSL)	<a href="#">mod_proxy_connect</a>
FastCGI	<a href="#">mod_proxy_fcgi</a>
ftp	<a href="#">mod_proxy_ftp</a>
HTTP/0.9, HTTP/1.0, and HTTP/1.1	<a href="#">mod_proxy_http</a>
SCGI	<a href="#">mod_proxy_scgi</a>
WS and WSS (Web-sockets)	<a href="#">mod_proxy_wstunnel</a>

In addition, extended features are provided by other modules. Caching is provided by [mod\\_cache](#) and related modules. The ability to contact remote servers using the SSL/TLS protocol is provided by the SSLProxy\* directives of [mod\\_ssl](#). These additional modules will need to be loaded and configured to take advantage of these features.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_cache](#)

[mod\\_proxy\\_ajp](#)

[mod\\_proxy\\_connect](#)

[mod\\_proxy\\_fcgi](#)

[mod\\_proxy\\_ftp](#)

[mod\\_proxy\\_http](#)

mod\_proxy scgi  
mod\_proxy wstunnel  
mod\_proxy balancer  
mod\_ssl



---

Apache HTTP Server can be configured in both a *forward* and *reverse proxy* (also known as *gateway*) mode.

An ordinary *forward proxy* is an intermediate server that sits between the client and the *origin server*. In order to get content from the origin server, the client sends a request to the proxy naming the origin server as the target. The proxy then requests the content from the origin server and returns it to the client. The client must be specially configured to use the forward proxy to access other sites.

A typical usage of a forward proxy is to provide Internet access to internal clients that are otherwise restricted by a firewall. The forward proxy can also use caching (as provided by [mod\\_cache](#)) to reduce network usage.

The forward proxy is activated using the [ProxyRequests](#) directive. Because forward proxies allow clients to access arbitrary sites through your server and to hide their true origin, it is essential that you [secure your server](#) so that only authorized clients can access the proxy before activating a forward proxy.

A *reverse proxy* (or *gateway*), by contrast, appears to the client just like an ordinary web server. No special configuration on the client is necessary. The client makes ordinary requests for content in the namespace of the reverse proxy. The reverse proxy then decides where to send those requests and returns the content as if it were itself the origin.

A typical usage of a reverse proxy is to provide Internet users access to a server that is behind a firewall. Reverse proxies can also be used to balance load among several back-end servers or to provide caching for a slower back-end server. In addition, reverse proxies can be used simply to bring several servers into

the same URL space.

A reverse proxy is activated using the [ProxyPass](#) directive or the [P] flag to the [RewriteRule](#) directive. It is **not** necessary to turn [ProxyRequests](#) on in order to configure a reverse proxy.



## Basic Examples

The examples below are only a very basic idea to help you get started. Please read the documentation on the individual directives.

In addition, if you wish to have caching enabled, consult the documentation from [mod\\_cache](#).

### Reverse Proxy

```
ProxyPass "/foo" "http://foo.example.com/bar"  
ProxyPassReverse "/foo" "http://foo.example.com/bar"
```

### Forward Proxy

```
ProxyRequests On  
ProxyVia On  
  
<Proxy "*">  
    Require host internal.example.com  
</Proxy>
```



You can also force a request to be handled as a reverse-proxy request, by creating a suitable Handler pass-through. The example configuration below will pass all requests for PHP scripts to the specified FastCGI server using reverse proxy:

### Reverse Proxy PHP scripts

```
<FilesMatch "\.php$">  
    # Unix sockets require 2.4.7 or later  
    SetHandler "proxy:unix:/path/to/app.sock|fcgi://localhost/"  
</FilesMatch>
```

This feature is available in Apache HTTP Server 2.4.10 and later.



The proxy manages the configuration of origin servers and their communication parameters in objects called *workers*. There are two built-in workers: the default forward proxy worker and the default reverse proxy worker. Additional workers can be configured explicitly.

The two default workers have a fixed configuration and will be used if no other worker matches the request. They do not use HTTP Keep-Alive or connection reuse. The TCP connections to the origin server will instead be opened and closed for each request.

Explicitly configured workers are identified by their URL. They are usually created and configured using [ProxyPass](#) or [ProxyPassMatch](#) when used for a reverse proxy:

```
ProxyPass "/example" "http://backend.example.com"
```

This will create a worker associated with the origin server URL `http://backend.example.com` that will use the given timeout values. When used in a forward proxy, workers are usually defined via the [ProxySet](#) directive:

```
ProxySet "http://backend.example.com" connectiontimeout=5 timeout=30
```

or alternatively using [Proxy](#) and [ProxySet](#):

```
<Proxy "http://backend.example.com">  
  ProxySet connectiontimeout=5 timeout=30  
</Proxy>
```

Using explicitly configured workers in the forward mode is not very

common, because forward proxies usually communicate with many different origin servers. Creating explicit workers for some of the origin servers can still be useful if they are used very often. Explicitly configured workers have no concept of forward or reverse proxying by themselves. They encapsulate a common concept of communication with origin servers. A worker created by [ProxyPass](#) for use in a reverse proxy will also be used for forward proxy requests whenever the URL to the origin server matches the worker URL, and vice versa.

The URL identifying a direct worker is the URL of its origin server including any path components given:

```
ProxyPass "/examples" "http://backend.example.com/"
ProxyPass "/docs" "http://backend.example.com/docs"
```

This example defines two different workers, each using a separate connection pool and configuration.

### Worker Sharing

Worker sharing happens if the worker URLs overlap, which occurs when the URL of some worker is a leading substring of the URL of another worker defined later in the configuration file. In the following example

```
ProxyPass "/apps" "http://backend.example.com/" timeout=60
ProxyPass "/examples" "http://backend.example.com/examples"
```

the second worker isn't actually created. Instead the first worker is used. The benefit is, that there is only one connection pool, so connections are more often reused. Note that all configuration attributes given explicitly for the later worker will be ignored. This will be logged as a warning. In the above

example, the resulting timeout value for the URL `/examples` will be 60 instead of 10!

If you want to avoid worker sharing, sort your worker definitions by URL length, starting with the longest worker URLs. If you want to maximize worker sharing, use the reverse sort order. See also the related warning about ordering [ProxyPass](#) directives.

Explicitly configured workers come in two flavors: *direct workers* and (*load*) *balancer workers*. They support many important configuration attributes which are described below in the [ProxyPass](#) directive. The same attributes can also be set using [ProxySet](#).

The set of options available for a direct worker depends on the protocol which is specified in the origin server URL. Available protocols include `ajp`, `fcgi`, `ftp`, `http` and `scgi`.

Balancer workers are virtual workers that use direct workers known as their members to actually handle the requests. Each balancer can have multiple members. When it handles a request, it chooses a member based on the configured load balancing algorithm.

A balancer worker is created if its worker URL uses `balancer` as the protocol scheme. The balancer URL uniquely identifies the balancer worker. Members are added to a balancer using [BalancerMember](#).

### **DNS resolution for origin domains**

DNS resolution happens when the socket to the origin domain is created for the first time. When connection reuse is enabled, each backend domain is resolved only once per child process,

and cached for all further connections until the child is recycled. This information should to be considered while planning DNS maintenance tasks involving backend domains. Please also check [ProxyPass](#) parameters for more details about connection reuse.



You can control who can access your proxy via the [<Proxy>](#) control block as in the following example:

```
<Proxy "*" >
  Require ip 192.168.0
</Proxy>
```

For more information on access control directives, see [mod\\_authz\\_host](#).

Strictly limiting access is essential if you are using a forward proxy (using the [ProxyRequests](#) directive). Otherwise, your server can be used by any client to access arbitrary hosts while hiding his or her true identity. This is dangerous both for your network and for the Internet at large. When using a reverse proxy (using the [ProxyPass](#) directive with `ProxyRequests Off`), access control is less critical because clients can only contact the hosts that you have specifically configured.

**See Also** the [Proxy-Chain-Auth](#) environment variable.



If you're using the [ProxyBlock](#) directive, hostnames' IP addresses are looked up and cached during startup for later match test. This may take a few seconds (or more) depending on the speed with which the hostname lookups occur.



## Intranet Proxy

An Apache httpd proxy server situated in an intranet needs to forward external requests through the company's firewall (for this, configure the [ProxyRemote](#) directive to forward the respective *scheme* to the firewall proxy). However, when it has to access resources within the intranet, it can bypass the firewall when accessing hosts. The [NoProxy](#) directive is useful for specifying which hosts belong to the intranet and should be accessed directly.

Users within an intranet tend to omit the local domain name from their WWW requests, thus requesting "http://somehost/" instead of `http://somehost.example.com/`. Some commercial proxy servers let them get away with this and simply serve the request, implying a configured local domain. When the [ProxyDomain](#) directive is used and the server is [configured for proxy service](#), Apache httpd can return a redirect response and send the client to the correct, fully qualified, server address. This is the preferred method since the user's bookmark files will then contain fully qualified hosts.



For circumstances where `mod_proxy` is sending requests to an origin server that doesn't properly implement keepalives or HTTP/1.1, there are two [environment variables](#) that can force the request to use HTTP/1.0 with no keepalive. These are set via the [SetEnv](#) directive.

These are the `force-proxy-request-1.0` and `proxy-nokeepalive` notes.

```
<Location "/buggyappserver/">
  ProxyPass "http://buggyappserver:7001/foo"
  SetEnv force-proxy-request-1.0 1
  SetEnv proxy-nokeepalive 1
</Location>
```

In 2.4.26 and later, the "no-proxy" environment variable can be set to disable `mod_proxy` processing the current request. This variable should be set with [SetEnvIf](#), as [SetEnv](#) is not evaluated early enough.



## Request Bodies

Some request methods such as POST include a request body. The HTTP protocol requires that requests which include a body either use chunked transfer encoding or send a Content-Length request header. When passing these requests on to the origin server, [mod\\_proxy\\_http](#) will always attempt to send the Content-Length. But if the body is large and the original request used chunked encoding, then chunked encoding may also be used in the upstream request. You can control this selection using [environment variables](#). Setting `proxy-sendcl` ensures maximum compatibility with upstream servers by always sending the Content-Length, while setting `proxy-sendchunked` minimizes resource usage by using chunked encoding.

Under some circumstances, the server must spool request bodies to disk to satisfy the requested handling of request bodies. For example, this spooling will occur if the original body was sent with chunked encoding (and is large), but the administrator has asked for backend requests to be sent with Content-Length or as HTTP/1.0. This spooling can also occur if the request body already has a Content-Length header, but the server is configured to filter incoming request bodies.

[LimitRequestBody](#) only applies to request bodies that the server will spool to disk



When acting in a reverse-proxy mode (using the [ProxyPass](#) directive, for example), [mod\\_proxy\\_http](#) adds several request headers in order to pass information to the origin server. These headers are:

### **X-Forwarded-For**

The IP address of the client.

### **X-Forwarded-Host**

The original host requested by the client in the Host HTTP request header.

### **X-Forwarded-Server**

The hostname of the proxy server.

Be careful when using these headers on the origin server, since they will contain more than one (comma-separated) value if the original request already contained one of these headers. For example, you can use `%{X-Forwarded-For}i` in the log format string of the origin server to log the original clients IP address, but you may get more than one address if the request passes through several proxies.

See also the [ProxyPreserveHost](#) and [ProxyVia](#) directives, which control other request headers.

Note: If you need to specify custom request headers to be added to the forwarded request, use the [RequestHeader](#) directive.



<b>Description:</b>	Number of additional Balancers that can be added Post-configuration
<b>Syntax:</b>	BalancerGrowth #
<b>Default:</b>	BalancerGrowth 5
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	BalancerGrowth is only available in Apache HTTP Server 2.3.13 and later.

This directive allows for growth potential in the number of Balancers available for a virtualhost in addition to the number pre-configured. It only takes effect if there is at least one pre-configured Balancer.



<b>Description:</b>	Inherit ProxyPassed Balancers/Workers from the main server
<b>Syntax:</b>	BalancerInherit On Off
<b>Default:</b>	BalancerInherit On
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	BalancerInherit is only available in Apache HTTP Server 2.4.5 and later.

This directive will cause the current server/vhost to "inherit" ProxyPass Balancers and Workers defined in the main server. This can cause issues and inconsistent behavior if using the Balancer Manager and so should be disabled if using that feature.

The setting in the global server defines the default for all vhosts.



<b>Description:</b>	Add a member to a load balancing group
<b>Syntax:</b>	<code>BalancerMember [<i>balancerurl</i>] <i>url</i> [<i>key=value</i> [<i>key=value</i> ...]]</code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	BalancerMember is only available in Apache HTTP Server 2.2 and later.

This directive adds a member to a load balancing group. It can be used within a `<Proxy balancer://...>` container directive and can take any of the key value pair parameters available to [ProxyPass](#) directives.

One additional parameter is available only to [BalancerMember](#) directives: *loadfactor*. This is the member load factor - a decimal number between 1.0 (default) and 100.0, which defines the weighted load to be applied to the member in question.

The *balancerurl* is only needed when not within a `<Proxy balancer://...>` container directive. It corresponds to the url of a balancer defined in [ProxyPass](#) directive.

The path component of the balancer URL in any `<Proxy balancer://...>` container directive is ignored.

Trailing slashes should typically be removed from the URL of a [BalancerMember](#).



<b>Description:</b>	Attempt to persist changes made by the Balancer Manager across restarts.
<b>Syntax:</b>	BalancerPersist On Off
<b>Default:</b>	BalancerPersist Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	BalancerPersist is only available in Apache HTTP Server 2.4.4 and later.

This directive will cause the shared memory storage associated with the balancers and balancer members to be persisted across restarts. This allows these local changes to not be lost during the normal restart/graceful state transitions.



<b>Description:</b>	Hosts, domains, or networks that will be connected to directly
<b>Syntax:</b>	NoProxy <i>host</i> [ <i>host</i> ] ...
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is only useful for Apache httpd proxy servers within intranets. The **NoProxy** directive specifies a list of subnets, IP addresses, hosts and/or domains, separated by spaces. A request to a host which matches one or more of these is always served directly, without forwarding to the configured **ProxyRemote** proxy server(s).

### Example

```
ProxyRemote    "*"    "http://firewall.example.com:81"
NoProxy       ".example.com" "192.168.112.0/21"
```

The *host* arguments to the **NoProxy** directive are one of the following type list:

### Domain

A *Domain* is a partially qualified DNS domain name, preceded by a period. It represents a list of hosts which logically belong to the same DNS domain or zone (*i.e.*, the suffixes of the hostnames are all ending in *Domain*).

### Examples

```
.com .example.org.
```

To distinguish *Domains* from **Hostnames** (both syntactically and semantically; a DNS domain can have a DNS A record,

too!), *Domains* are always written with a leading period.

### Note

Domain name comparisons are done without regard to the case, and *Domains* are always assumed to be anchored in the root of the DNS tree; therefore, the two domains `.Example.com` and `.example.com.` (note the trailing period) are considered equal. Since a domain comparison does not involve a DNS lookup, it is much more efficient than subnet comparison.

### SubNet

A *SubNet* is a partially qualified internet address in numeric (dotted quad) form, optionally followed by a slash and the netmask, specified as the number of significant bits in the *SubNet*. It is used to represent a subnet of hosts which can be reached over a common network interface. In the absence of the explicit net mask it is assumed that omitted (or zero valued) trailing digits specify the mask. (In this case, the netmask can only be multiples of 8 bits wide.) Examples:

#### **192.168 or 192.168.0.0**

the subnet 192.168.0.0 with an implied netmask of 16 valid bits (sometimes used in the netmask form 255.255.0.0)

#### **192.168.112.0/21**

the subnet 192.168.112.0/21 with a netmask of 21 valid bits (also used in the form 255.255.248.0)

As a degenerate case, a *SubNet* with 32 valid bits is the equivalent to an [IPAddr](#), while a *SubNet* with zero valid bits (e.g., 0.0.0.0/0) is the same as the constant `_Default_`, matching any IP address.

## ***IPAddr***

A *IPAddr* represents a fully qualified internet address in numeric (dotted quad) form. Usually, this address represents a host, but there need not necessarily be a DNS domain name connected with the address.

### **Example**

192.168.123.7

### **Note**

An *IPAddr* does not need to be resolved by the DNS system, so it can result in more effective apache performance.

## ***Hostname***

A *Hostname* is a fully qualified DNS domain name which can be resolved to one or more [IPAddrs](#) via the DNS domain name service. It represents a logical host (in contrast to [Domains](#), see above) and must be resolvable to at least one [IPAddr](#) (or often to a list of hosts with different [IPAddrs](#)).

### **Examples**

prep.ai.example.edu  
www.example.org

### **Note**

In many situations, it is more effective to specify an [IPAddr](#) in place of a *Hostname* since a DNS lookup can be avoided. Name resolution in Apache httpd can take a remarkable deal of time when the connection to the name server uses a slow PPP link.

*Hostname* comparisons are done without regard to the

case, and *Hostnames* are always assumed to be anchored in the root of the DNS tree; therefore, the two hosts `WWW.ExAmPle.com` and `www.example.com.` (note the trailing period) are considered equal.

## See also

- [DNS Issues](#)



## PROXY DIRECTIVES

<b>Description:</b>	Container for directives applied to proxied resources
<b>Syntax:</b>	<code>&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

Directives placed in `<Proxy>` sections apply only to matching proxied content. Shell-style wildcards are allowed.

For example, the following will allow only hosts in `yournetwork.example.com` to access content via your proxy server:

```
<Proxy "*">
    Require host yournetwork.example.com
</Proxy>
```

The following example will process all files in the `foo` directory of `example.com` through the `INCLUDES` filter when they are sent through the proxy server:

```
<Proxy "http://example.com/foo/*">
    SetOutputFilter INCLUDES
</Proxy>
```

### Differences from the Location configuration section

A backend URL matches the configuration section if it begins with the `wildcard-url` string, even if the last path segment in the directive only matches a prefix of the backend URL. For example, `<Proxy "http://example.com/foo">` matches all of `http://example.com/foo`, `http://example.com/foo/bar`, and

http://example.com/foobar. The matching of the final URL differs from the behavior of the [<Location>](#) section, which for purposes of this note treats the final path component as if it ended in a slash.

For more control over the matching, see [<ProxyMatch>](#).

## See also

- [<ProxyMatch>](#)



## ProxyAddHeaders Directive

<b>Description:</b>	Add proxy information in X-Forwarded-* headers
<b>Syntax:</b>	ProxyAddHeaders Off On
<b>Default:</b>	ProxyAddHeaders On
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in version 2.3.10 and later

This directive determines whether or not proxy related information should be passed to the backend server through X-Forwarded-For, X-Forwarded-Host and X-Forwarded-Server HTTP headers.

**Effectiveness**

This option is of use only for HTTP proxying, as handled by [mod\\_proxy\\_http](#).



## ProxyBadHeader Directive

<b>Description:</b>	Determines how to handle bad header lines in a response
<b>Syntax:</b>	ProxyBadHeader IsError   Ignore   StartBody
<b>Default:</b>	ProxyBadHeader IsError
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The **ProxyBadHeader** directive determines the behavior of **mod\_proxy** if it receives syntactically invalid response header lines (*i.e.* containing no colon) from the origin server. The following arguments are possible:

### **IsError**

Abort the request and end up with a 502 (Bad Gateway) response. This is the default behavior.

### **Ignore**

Treat bad header lines as if they weren't sent.

### **StartBody**

When receiving the first bad header line, finish reading the headers and treat the remainder as body. This helps to work around buggy backend servers which forget to insert an empty line between the headers and the body.



**Description:** Words, hosts, or domains that are banned from being proxied

**Syntax:** ProxyBlock \* |*word*|*host*|*domain*  
[*word*|*host*|*domain*] ...

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_proxy

The **ProxyBlock** directive specifies a list of words, hosts and/or domains, separated by spaces. HTTP, HTTPS, and FTP document requests to sites whose names contain matched words, hosts or domains are *blocked* by the proxy server. The proxy module will also attempt to determine IP addresses of list items which may be hostnames during startup, and cache them for match test as well. That may slow down the startup time of the server.

### Example

```
ProxyBlock "news.example.com" "auctions.example.com" "friends.e;
```

Note that example would also be sufficient to match any of these sites.

Hosts would also be matched if referenced by IP address.

Note also that

```
ProxyBlock "*"
```

blocks connections to all sites.



## ProxyDomain Directive

<b>Description:</b>	Default domain name for proxied requests
<b>Syntax:</b>	ProxyDomain <i>Domain</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is only useful for Apache httpd proxy servers within intranets. The `ProxyDomain` directive specifies the default domain which the apache proxy server will belong to. If a request to a host without a domain name is encountered, a redirection response to the same host with the configured *Domain* appended will be generated.

**Example**

```
ProxyRemote    "*"    "http://firewall.example.com:81"  
NoProxy       ".example.com" "192.168.112.0/21"  
ProxyDomain   ".example.com"
```



<b>Description:</b>	Override error pages for proxied content
<b>Syntax:</b>	ProxyErrorOverride On Off
<b>Default:</b>	ProxyErrorOverride Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is useful for reverse-proxy setups where you want to have a common look and feel on the error pages seen by the end user. This also allows for included files (via [mod\\_include](#)'s SSI) to get the error code and act accordingly. (Default behavior would display the error page of the proxied server. Turning this on shows the SSI Error message.)

This directive does not affect the processing of informational (1xx), normal success (2xx), or redirect (3xx) responses.



<b>Description:</b>	Determine size of internal data throughput buffer
<b>Syntax:</b>	ProxyIOBufferSize <i>bytes</i>
<b>Default:</b>	ProxyIOBufferSize 8192
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `ProxyIOBufferSize` directive adjusts the size of the internal buffer which is used as a scratchpad for the data between input and output. The size must be at least 512.

In almost every case, there's no reason to change that value.

If used with AJP, this directive sets the maximum AJP packet size in bytes. Values larger than 65536 are set to 65536. If you change it from the default, you must also change the `packetSize` attribute of your AJP connector on the Tomcat side! The attribute `packetSize` is only available in Tomcat 5.5.20+ and 6.0.2+

Normally it is not necessary to change the maximum packet size. Problems with the default value have been reported when sending certificates or certificate chains.



<b>Description:</b>	Container for directives applied to regular-expression-matched proxied resources
<b>Syntax:</b>	<code>&lt;ProxyMatch regex&gt; ...&lt;/ProxyMatch&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `<ProxyMatch>` directive is identical to the `<Proxy>` directive, except that it matches URLs using [regular expressions](#).

From 2.4.8 onwards, named groups and backreferences are captured and written to the environment with the corresponding name prefixed with "MATCH\_" and in upper case. This allows elements of URLs to be referenced from within [expressions](#) and modules like `mod_rewrite`. In order to prevent confusion, numbered (unnamed) backreferences are ignored. Use named groups instead.

```
<ProxyMatch "^http://(?<sitename>[^/]+)">
    Require ldap-group cn=%{env:MATCH_SITENAME}
</ProxyMatch>
```

## See also

- [<Proxy>](#)



## ProxyMaxForwards Directive

<b>Description:</b>	Maximum number of proxies that a request can be forwarded through
<b>Syntax:</b>	ProxyMaxForwards <i>number</i>
<b>Default:</b>	ProxyMaxForwards -1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Default behaviour changed in 2.2.7

The **ProxyMaxForwards** directive specifies the maximum number of proxies through which a request may pass if there's no Max-Forwards header supplied with the request. This may be set to prevent infinite proxy loops or a DoS attack.

### Example

```
ProxyMaxForwards 15
```

Note that setting **ProxyMaxForwards** is a violation of the HTTP/1.1 protocol (RFC2616), which forbids a Proxy setting Max-Forwards if the Client didn't set it. Earlier Apache httpd versions would always set it. A negative **ProxyMaxForwards** value, including the default -1, gives you protocol-compliant behavior but may leave you open to loops.



## ProxyPass Directive

<b>Description:</b>	Maps remote servers into the local server URL-space
<b>Syntax:</b>	ProxyPass <i>[path]</i> <i>! url</i> [ <i>key=value</i> [ <i>key=value ...</i> ]] [ <i>nocanon</i> ] [ <i>interpolate</i> ] [ <i>noquery</i> ]
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Unix Domain Socket (UDS) support added in 2.4.7

This directive allows remote servers to be mapped into the space of the local server. The local server does not act as a proxy in the conventional sense but appears to be a mirror of the remote server. The local server is often called a *reverse proxy* or *gateway*. The *path* is the name of a local virtual path; *url* is a partial URL for the remote server and cannot include a query string.

It is strongly suggested to review the concept of a [Worker](#) before proceeding any further with this section.

This directive is not supported within [<Directory>](#) and [<Files>](#) containers.

The [ProxyRequests](#) directive should usually be set **off** when using [ProxyPass](#).

In 2.4.7 and later, support for using a Unix Domain Socket is available by using a target which prepends `unix:/path/lis.sock|`. For example, to proxy HTTP and target the UDS at `/home/www/socket`, you would use

unix:/home/www.socket|http://localhost/whatever/.

**Note:** The path associated with the unix: URL is `DefaultRuntimeDir` aware.

When used inside a `<Location>` section, the first argument is omitted and the local directory is obtained from the `<Location>`. The same will occur inside a `<LocationMatch>` section; however, ProxyPass does not interpret the regexp as such, so it is necessary to use `ProxyPassMatch` in this situation instead.

Suppose the local server has address `http://example.com/`; then

```
<Location "/mirror/foo/">  
    ProxyPass "http://backend.example.com/"  
</Location>
```

will cause a local request for `http://example.com/mirror/foo/bar` to be internally converted into a proxy request to `http://backend.example.com/bar`.

If you require a more flexible reverse-proxy configuration, see the `RewriteRule` directive with the [P] flag.

The following alternative syntax is possible; however, it can carry a performance penalty when present in very large numbers. The advantage of the below syntax is that it allows for dynamic control via the `Balancer Manager` interface:

```
ProxyPass "/mirror/foo/" "http://backend.exa
```

If the first argument ends with a trailing `/`, the second argument should also end with a trailing `/`, and vice versa. Otherwise, the resulting requests to the backend may miss some needed slashes and do not deliver the expected results.

The `!` directive is useful in situations where you don't want to reverse-proxy a subdirectory, e.g.

```
<Location "/mirror/foo/">
    ProxyPass "http://backend.example.com/"
</Location>
<Location "/mirror/foo/i">
    ProxyPass "!"
</Location>
```

```
ProxyPass "/mirror/foo/i" "!"
ProxyPass "/mirror/foo" "http://backend.exar
```

will proxy all requests to `/mirror/foo` to `backend.example.com` *except* requests made to `/mirror/foo/i`.

### Ordering ProxyPass Directives

The configured `ProxyPass` and `ProxyPassMatch` rules are checked in the order of configuration. The first rule that matches wins. So usually you should sort conflicting `ProxyPass` rules starting with the longest URLs first. Otherwise, later rules for longer URLs will be hidden by any earlier rule which uses a leading substring of the URL. Note that there is some relation with worker sharing. In contrast, only one `ProxyPass` directive can be placed in a `Location` block, and the most specific

location will take precedence.

For the same reasons, exclusions must come *before* the general `ProxyPass` directives. In 2.4.26 and later, the "no-proxy" environment variable is an alternative to exclusions, and is the only way to configure an exclusion of a `ProxyPass` directive in `Location` context. This variable should be set with `SetEnvIf`, as `SetEnv` is not evaluated early enough.

## ProxyPass key=value Parameters

In Apache HTTP Server 2.1 and later, `mod_proxy` supports pooled connections to a backend server. Connections created on demand can be retained in a pool for future use. Limits on the pool size and other settings can be coded on the `ProxyPass` directive using `key=value` parameters, described in the tables below.

### Maximum connections to the backend

By default, `mod_proxy` will allow and retain the maximum number of connections that could be used simultaneously by that web server child process. Use the `max` parameter to reduce the number from the default. The pool of connections is maintained per web server child process, and `max` and other settings are not coordinated among all child processes, except when only one child process is allowed by configuration or MPM design.

Use the `ttl` parameter to set an optional time to live; connections which have been unused for at least `ttl` seconds will be closed. `ttl` can be used to avoid using a connection which is subject to closing because of the backend server's keep-alive timeout.

### Example

```
ProxyPass "/example" "http://backend.example.com" max=20 ttl=120
```

## Worker|BalancerMember parameters

Parameter	Default	Description
min	0	Minimum number of connection pool entries, unrelated to the actual number of connections. This only needs to be modified from the default for special circumstances where heap memory associated with the backend connections should be preallocated or retained.
max	1...n	Maximum number of connections that will be allowed to the backend server. The default for this limit is the number of threads per process in the active MPM. In the Prefork MPM, this is always 1, while with other MPMs, it is controlled by the <b>ThreadsPerChild</b> directive.
smax	max	Retained connection pool entries above this limit are freed during certain operations if they have been unused for longer than the time to live, controlled by the <code>ttl</code> parameter. If the

acquire	-	<p>connection pool entry has an associated connection, it will be closed. This only needs to be modified from the default for special circumstances where connection pool entries and any associated connections which have exceeded the time to live need to be freed or closed more aggressively.</p> <p>If set, this will be the maximum time to wait for a free connection in the connection pool, in milliseconds. If there are no free connections in the pool, the Apache httpd will return SERVER_BUSY status to the client.</p>
connectiontimeout	timeout	<p>Connect timeout in seconds. The number of seconds Apache httpd waits for the creation of a connection to the backend to complete. By adding a postfix of ms, the timeout can be also set in milliseconds.</p>
disablereuse	Off	<p>This parameter should be used when you want to force mod_proxy to immediately close a connection to the backend after being used, and thus, disable its</p>

persistent connection and pool for that backend. This helps in various situations where a firewall between Apache httpd and the backend server (regardless of protocol) tends to silently drop connections or when backends themselves may be under round-robin DNS. When connection reuse is enabled each backend domain is resolved (with a DNS query) only once per child process and cached for all further connections until the child is recycled. To disable connection reuse, set this property value to On.

enablereuse      On

This is the inverse of 'disablereuse' above, provided as a convenience for scheme handlers that require opt-in for connection reuse (such as [mod\\_proxy\\_fcgi](#)). 2.4.11 and later only.

flushpackets      off

Determines whether the proxy module will auto-flush the output brigade after each "chunk" of data. 'off' means that it will flush only when needed; 'on' means after each chunk is sent; and 'auto'

		means poll/wait for a period of time and flush if no input has been received for 'flushwait' milliseconds. Currently, this is in effect only for AJP.
flushwait	10	The time to wait for additional input, in milliseconds, before flushing the output brigade if 'flushpackets' is 'auto'.
jobuffersize	8192	Adjusts the size of the internal scratchpad IO buffer. This allows you to override the <code>ProxyIOBufferSize</code> for a specific worker. This must be at least 512 or set to 0 for the system default of 8192.
keepalive	Off	This parameter should be used when you have a firewall between your Apache httpd and the backend server, which tends to drop inactive connections. This flag will tell the Operating System to send <code>KEEP_ALIVE</code> messages on inactive connections and thus prevent the firewall from dropping the connection. To enable keepalive, set this property value to On.  The frequency of initial and

subsequent TCP keepalive probes depends on global OS settings, and may be as high as 2 hours. To be useful, the frequency configured in the OS must be smaller than the threshold used by the firewall.

lbset	0	Sets the load balancer cluster set that the worker is a member of. The load balancer will try all members of a lower numbered lbset before trying higher numbered ones.
ping	0	Ping property tells the webserver to "test" the connection to the backend before forwarding the request. For AJP, it causes <a href="#">mod_proxy_ajp</a> to send a CPING request on the ajp13 connection (implemented on Tomcat 3.3.2+, 4.1.28+ and 5.0.13+). For HTTP, it causes <a href="#">mod_proxy_http</a> to send a 100-Continue to the backend (only valid for HTTP/1.1 - for non HTTP/1.1 backends, this property has no effect). In both cases, the parameter is the delay in seconds to wait for the reply.

receivebuffersize	0	<p>This feature has been added to avoid problems with hung and busy backends. This will increase the network traffic during the normal operation which could be an issue, but it will lower the traffic in case some of the cluster nodes are down or busy. By adding a postfix of ms, the delay can be also set in milliseconds.</p>
		<p>Adjusts the size of the explicit (TCP/IP) network buffer size for proxied connections. This allows you to override the <code>ProxyReceiveBufferSize</code> for a specific worker. This must be at least 512 or set to 0 for the system default.</p>
redirect	-	<p>Redirection Route of the worker. This value is usually set dynamically to enable safe removal of the node from the cluster. If set, all requests without session id will be redirected to the BalancerMember that has route parameter equal to this value.</p>
retry	60	<p>Connection pool worker retry timeout in seconds. If the connection pool worker to the backend server is in the error state, Apache httpd will not</p>

forward any requests to that server until the timeout expires. This enables to shut down the backend server for maintenance and bring it back online later. A value of 0 means always retry workers in an error state with no timeout.

route -

Route of the worker when used inside load balancer. The route is a value appended to session id.

status -

Single letter value defining the initial status of this worker.

D: Worker is disabled and will not accept any requests.

S: Worker is administratively stopped.

I: Worker is in ignore-errors mode and will always be considered available.

H: Worker is in hot-standby mode and will only be used if no other viable workers are available.

E: Worker is in an error state.

N: Worker is in drain mode and will only accept existing sticky sessions destined for itself and ignore all other

requests.

Status can be set (which is the default) by prepending with '+' or cleared by prepending with '-'. Thus, a setting of 'S-E' sets this worker to Stopped and clears the in-error flag.

timeout	<a href="#">ProxyTimeout</a>	Connection timeout in seconds. The number of seconds Apache httpd waits for data sent by / to the backend.
ttl	-	Time to live for inactive connections and associated connection pool entries, in seconds. Once reaching this limit, a connection will not be used again; it will be closed at some later time.
flusher	flush	Name of the provider used by <a href="#">mod_proxy_fdpass</a> . See the documentation of this module for more details.
secret	-	Value of secret used by <a href="#">mod_proxy_ajp</a> . See the documentation of this module for more details.
upgrade	WebSocket	Protocol accepted in the Upgrade header by <a href="#">mod_proxy_wstunnel</a> . See

the documentation of this module for more details.

If the Proxy directive scheme starts with the balancer:// (eg: balancer://cluster, any path information is ignored), then a virtual worker that does not really communicate with the backend server will be created. Instead, it is responsible for the management of several "real" workers. In that case, the special set of parameters can be added to this virtual worker. See [mod\\_proxy\\_balancer](#) for more information about how the balancer works.

### Balancer parameters

Parameter	Default	Description
lbmethod	byrequests	Balancer load-balance method. Select the load-balancing scheduler method to use. Either byrequests, to perform weighted request counting; bytraffic to perform weighted traffic byte count balancing; or bybusyness, to perform pending request balancing. The default is byrequests.
maxattempts	One less than the number of workers, or 1 with a single worker.	Maximum number of failover attempts before giving up.
nofailover	Off	If set to On, the session will break if the worker is in error state or disabled. S

stickysession	-	<p>this value to On if backend servers do not support session replication.</p> <p>Balancer sticky session name. The value is usually set to something like JSESSIONID or PHPSESSIONID, and depends on the backend application server that supports sessions. If the backend application server uses a different name for cookies and URL encoded id (like servlet containers) to separate them. The first part is for the cookie and the second for the path. Available in Apache HTTP Server 2.0 and later.</p>
stickysessionsep	","	<p>Sets the separation symbol in the session cookie. Some backend application servers do not use the '.' symbol. For example, the Oracle Weblogic server uses '!'. The correct symbol can be set using this option. A setting of 'Off' signifies that no symbol is used.</p>
semicolonpathdelim	Off	<p>If set to On, the semi-colon character will be used as an additional sticky session path delimiter/separator. This is mainly used to emulate mod_jk's behavior when dealing with paths such as</p> <p>JSESSIONID=6736bcf34;foo=aa</p>
timeout	0	<p>Balancer timeout in seconds. If set, this will be the maximum time to wait for a free worker. The default is to not wait.</p>
failonstatus	-	<p>A single or comma-separated list of</p>

		<p>HTTP status codes. If set, this will force the worker into error state when the backend returns any status code in the list. Worker recovery behaves the same as other worker errors.</p>
failontimeout	Off	<p>If set, an IO read timeout after a request is sent to the backend will force the worker into error state. Worker recovery behaves the same as other worker errors.</p> <p>Available in Apache HTTP Server 2.4 and later.</p>
nonce	<auto>	<p>The protective nonce used in the balancer-manager application page. The default is to use an automatically determined UUID-based nonce, to provide for further protection for the page. If set, then the nonce is set to the value. A setting of None disables all nonce checking.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p><b>Note</b></p> <p>In addition to the nonce, the balancer-manager page should be protected via an ACL.</p> </div>
growth	0	<p>Number of additional BalancerMembers to allow to be added to this balancer in addition to those defined at configuration.</p>
forcerecovery	On	<p>Force the immediate recovery of all workers without considering the retry parameter of the workers if all workers</p>

of a balancer are in error state. There might be cases where an already overloaded backend can get into deeper trouble if the recovery of all workers is enforced without considering the retry parameter of each worker. In this case, set it to Off.

Available in Apache HTTP Server 2.4.18 and later.

A sample balancer setup:

```
ProxyPass "/special-area" "http://special.example.com/special-area"
ProxyPass "/" "balancer://mycluster/" sticky
<Proxy "balancer://mycluster">
    BalancerMember "ajp://1.2.3.4:8009" loadfactor=1
    BalancerMember "ajp://1.2.3.5:8009" loadfactor=1
    # Less powerful server, don't send as much traffic
    BalancerMember "ajp://1.2.3.6:8009" loadfactor=0.5
</Proxy>
```

Setting up a hot-standby that will only be used if no other members are available:

```
ProxyPass "/" "balancer://hotcluster/"
<Proxy "balancer://hotcluster">
    BalancerMember "ajp://1.2.3.4:8009" loadfactor=1
    BalancerMember "ajp://1.2.3.5:8009" loadfactor=1
    # The server below is on hot standby
    BalancerMember "ajp://1.2.3.6:8009" status=NotAlive
    ProxySet lbmethod=bytraffic
</Proxy>
```

**Additional ProxyPass Keywords**

Normally, `mod_proxy` will canonicalise `ProxyPassed` URLs. But this may be incompatible with some backends, particularly those that make use of `PATH_INFO`. The optional `nocanon` keyword suppresses this and passes the URL path "raw" to the backend. Note that this keyword may affect the security of your backend, as it removes the normal limited protection against URL-based attacks provided by the proxy.

Normally, `mod_proxy` will include the query string when generating the `SCRIPT_FILENAME` environment variable. The optional `noquery` keyword (available in `httpd 2.4.1` and later) prevents this.

The optional `interpolate` keyword, in combination with `ProxyPassInterpolateEnv`, causes the `ProxyPass` to interpolate environment variables, using the syntax `${VARIABLE}`. Note that many of the standard CGI-derived environment variables will not exist when this interpolation happens, so you may still have to resort to `mod_rewrite` for complex rules. Also note that interpolation is not supported within the scheme portion of a URL. Dynamic determination of the scheme can be accomplished with `mod_rewrite` as in the following example.

```
RewriteEngine On

RewriteCond "%{HTTPS}" =off
RewriteRule "." "-" [E=protocol:http]
RewriteCond "%{HTTPS}" =on
RewriteRule "." "-" [E=protocol:https]

RewriteRule "^/mirror/foo/(.*)" "%{ENV:protocol}://ba
ProxyPassReverse "/mirror/foo/" "http://ba
ProxyPassReverse "/mirror/foo/" "https://ba
```



## ProxyPassInherit Directive

<b>Description:</b>	Inherit ProxyPass directives defined from the main server
<b>Syntax:</b>	ProxyPassInherit On Off
<b>Default:</b>	ProxyPassInherit On
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	ProxyPassInherit is only available in Apache HTTP Server 2.4.5 and later.

This directive will cause the current server/vhost to "inherit" [ProxyPass](#) directives defined in the main server. This can cause issues and inconsistent behavior if using the Balancer Manager for dynamic changes and so should be disabled if using that feature.

The setting in the global server defines the default for all vhosts.

Disabling ProxyPassInherit also disables [BalancerInherit](#).



## ProxyPassInterpolateEnv Directive

<b>Description:</b>	Enable Environment Variable interpolation in Reverse Proxy configurations
<b>Syntax:</b>	ProxyPassInterpolateEnv On Off
<b>Default:</b>	ProxyPassInterpolateEnv Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in httpd 2.2.9 and later

This directive, together with the *interpolate* argument to [ProxyPass](#), [ProxyPassReverse](#), [ProxyPassReverseCookieDomain](#), and [ProxyPassReverseCookiePath](#), enables reverse proxies to be dynamically configured using environment variables which may be set by another module such as [mod\\_rewrite](#). It affects the [ProxyPass](#), [ProxyPassReverse](#), [ProxyPassReverseCookieDomain](#), and [ProxyPassReverseCookiePath](#) directives and causes them to substitute the value of an environment variable varname for the string `${varname}` in configuration directives if the *interpolate* option is set.

Keep this turned off (for server performance) unless you need it!



## ProxyPassMatch Directive

<b>Description:</b>	Maps remote servers into the local server URL-space using regular expressions
<b>Syntax:</b>	ProxyPassMatch [ <i>regex</i> ] !  <i>url</i> [ <i>key=value</i> [ <i>key=value</i> ...]]
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is equivalent to [ProxyPass](#) but makes use of regular expressions instead of simple prefix matching. The supplied regular expression is matched against the *url*, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a new *url*.

**Note:** This directive cannot be used within a <Directory> context.

Suppose the local server has address `http://example.com/`; then

```
ProxyPassMatch "^/(.*\.gif)$" "http://backer
```

will cause a local request for `http://example.com/foo/bar.gif` to be internally converted into a proxy request to `http://backend.example.com/foo/bar.gif`.

### Note

The URL argument must be parsable as a URL *before* regexp substitutions (as well as after). This limits the matches you can use. For instance, if we had used

```
ProxyPassMatch "^(/.*\.gif)$" "http://backend.example.com:"
```

in our previous example, it would fail with a syntax error at server startup. This is a bug (PR 46665 in the ASF bugzilla), and the workaround is to reformulate the match:

```
ProxyPassMatch "^/(.*\.gif)$" "http://backend.example.com:"
```

The ! directive is useful in situations where you don't want to reverse-proxy a subdirectory.

When used inside a [<LocationMatch>](#) section, the first argument is omitted and the regexp is obtained from the [<LocationMatch>](#).

If you require a more flexible reverse-proxy configuration, see the [RewriteRule](#) directive with the [P] flag.

### Default Substitution

When the URL parameter doesn't use any backreferences into the regular expression, the original URL will be appended to the URL parameter.

### Security Warning

Take care when constructing the target URL of the rule, considering the security impact from allowing the client influence over the set of URLs to which your server will act as a proxy. Ensure that the scheme and hostname part of the URL is either fixed or does not allow the client undue influence.



<b>Description:</b>	Adjusts the URL in HTTP response headers sent from a reverse proxied server
<b>Syntax:</b>	ProxyPassReverse [ <i>path</i> ] <i>url</i> [ <i>interpolate</i> ]
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive lets Apache httpd adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache httpd is used as a reverse proxy (or gateway) to avoid bypassing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy.

Only the HTTP response headers specifically mentioned above will be rewritten. Apache httpd will not rewrite other response headers, nor will it by default rewrite URL references inside HTML pages. This means that if the proxied content contains absolute URL references, they will bypass the proxy. To rewrite HTML content to match the proxy, you must load and enable [mod\\_proxy\\_html](#).

*path* is the name of a local virtual path; *url* is a partial URL for the remote server. These parameters are used the same way as for the [ProxyPass](#) directive.

For example, suppose the local server has address `http://example.com/`; then

```
ProxyPass          "/mirror/foo/" "http://bac
ProxyPassReverse  "/mirror/foo/" "http://bac
ProxyPassReverseCookieDomain "backend.examp
```

```
ProxyPassReverseCookiePath "/" "/mirror/fo
```

will not only cause a local request for the `http://example.com/mirror/foo/bar` to be internally converted into a proxy request to `http://backend.example.com/bar` (the functionality which ProxyPass provides here). It also takes care of redirects which the server `backend.example.com` sends when redirecting `http://backend.example.com/bar` to `http://backend.example.com/quux`. Apache httpd adjusts this to `http://example.com/mirror/foo/quux` before forwarding the HTTP redirect response to the client. Note that the hostname used for constructing the URL is chosen in respect to the setting of the [UseCanonicalName](#) directive.

Note that this [ProxyPassReverse](#) directive can also be used in conjunction with the proxy feature (`RewriteRule ... [P]`) from [mod\\_rewrite](#) because it doesn't depend on a corresponding [ProxyPass](#) directive.

The optional *interpolate* keyword, used together with [ProxyPassInterpolateEnv](#), enables interpolation of environment variables specified using the format `${VARIABLE}`. Note that interpolation is not supported within the scheme portion of a URL.

When used inside a [<Location>](#) section, the first argument is omitted and the local directory is obtained from the [<Location>](#). The same occurs inside a [<LocationMatch>](#) section, but will probably not work as intended, as ProxyPassReverse will interpret the regexp literally as a path; if needed in this situation, specify the ProxyPassReverse outside the section or in a separate [<Location>](#) section.

This directive is not supported in <Directory> or <Files> sections.



**Description:** Adjusts the Domain string in Set-Cookie headers from a reverse- proxied server

**Syntax:** ProxyPassReverseCookieDomain  
*internal-domain public-domain*  
*[interpolate]*

**Context:** server config, virtual host, directory

**Status:** Extension

**Module:** mod\_proxy

Usage is basically similar to [ProxyPassReverse](#), but instead of rewriting headers that are a URL, this rewrites the domain string in Set -Cookie headers.



<b>Description:</b>	Adjusts the Path string in Set-Cookie headers from a reverse- proxied server
<b>Syntax:</b>	<code>ProxyPassReverseCookiePath <i>internal-path</i> <i>public-path</i> [<i>interpolate</i>]</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

Useful in conjunction with [ProxyPassReverse](#) in situations where backend URL paths are mapped to public paths on the reverse proxy. This directive rewrites the path string in Set - Cookie headers. If the beginning of the cookie path matches *internal-path*, the cookie path will be replaced with *public-path*.

In the example given with [ProxyPassReverse](#), the directive:

```
ProxyPassReverseCookiePath "/" "/mirror/fo
```

will rewrite a cookie with backend path / (or /example or, in fact, anything) to /mirror/foo/.



## ProxyPreserveHost Directive

<b>Description:</b>	Use incoming Host HTTP request header for proxy request
<b>Syntax:</b>	ProxyPreserveHost On Off
<b>Default:</b>	ProxyPreserveHost Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Usable in directory context in 2.3.3 and later.

When enabled, this option will pass the Host: line from the incoming request to the proxied host, instead of the hostname specified in the [ProxyPass](#) line.

This option should normally be turned Off. It is mostly useful in special configurations like proxied mass name-based virtual hosting, where the original Host header needs to be evaluated by the backend server.



**Description:** Network buffer size for proxied HTTP and FTP connections

**Syntax:** ProxyReceiveBufferSize *bytes*

**Default:** ProxyReceiveBufferSize 0

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_proxy

The `ProxyReceiveBufferSize` directive specifies an explicit (TCP/IP) network buffer size for proxied HTTP and FTP connections, for increased throughput. It has to be greater than 512 or set to 0 to indicate that the system's default buffer size should be used.

### Example

```
ProxyReceiveBufferSize 2048
```



<b>Description:</b>	Remote proxy used to handle certain requests
<b>Syntax:</b>	ProxyRemote <i>match remote-server</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This defines remote proxies to this proxy. *match* is either the name of a URL-scheme that the remote server supports, or a partial URL for which the remote server should be used, or \* to indicate the server should be contacted for all requests. *remote-server* is a partial URL for the remote server. Syntax:

```
remote-server = scheme://hostname[:port]
```

*scheme* is effectively the protocol that should be used to communicate with the remote server; only http and https are supported by this module. When using https, the requests are forwarded through the remote proxy using the HTTP CONNECT method.

### Example

```
ProxyRemote "http://goodguys.example.com/" "http://mirrorguys.e  
ProxyRemote "*" "http://cleverproxy.localdomain"  
ProxyRemote "ftp" "http://ftpproxy.mydomain:8080"
```

In the last example, the proxy will forward FTP requests, encapsulated as yet another HTTP proxy request, to another proxy which can handle them.

This option also supports reverse proxy configuration; a backend webserver can be embedded within a virtualhost URL space even if that server is hidden by another forward proxy.



## ProxyRemoteMatch Directive

<b>Description:</b>	Remote proxy used to handle requests matched by regular expressions
<b>Syntax:</b>	<code>ProxyRemoteMatch <i>regex remote-server</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `ProxyRemoteMatch` is identical to the `ProxyRemote` directive, except that the first argument is a regular expression match against the requested URL.



## ProxyRequests Directive

<b>Description:</b>	Enables forward (standard) proxy requests
<b>Syntax:</b>	ProxyRequests On Off
<b>Default:</b>	ProxyRequests Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This allows or prevents Apache httpd from functioning as a forward proxy server. (Setting ProxyRequests to Off does not disable use of the [ProxyPass](#) directive.)

In a typical reverse proxy or gateway configuration, this option should be set to Off.

In order to get the functionality of proxying HTTP or FTP sites, you need also [mod\\_proxy\\_http](#) or [mod\\_proxy\\_ftp](#) (or both) present in the server.

In order to get the functionality of (forward) proxying HTTPS sites, you need [mod\\_proxy\\_connect](#) enabled in the server.

### Warning

Do not enable proxying with [ProxyRequests](#) until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

### See also

- [Forward and Reverse Proxies/Gateways](#)



<b>Description:</b>	Set various Proxy balancer or member parameters
<b>Syntax:</b>	<code>ProxySet <i>url</i> <i>key=value</i> [<i>key=value</i> ...]</code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	ProxySet is only available in Apache HTTP Server 2.2 and later.

This directive is used as an alternate method of setting any of the parameters available to Proxy balancers and workers normally done via the [ProxyPass](#) directive. If used within a `<Proxy balancer url|worker url>` container directive, the *url* argument is not required. As a side effect the respective balancer or worker gets created. This can be useful when doing reverse proxying via a [RewriteRule](#) instead of a [ProxyPass](#) directive.

```
<Proxy "balancer://hotcluster">
  BalancerMember "http://www2.example.com:8080" loadfactor=1
  BalancerMember "http://www3.example.com:8080" loadfactor=2
  ProxySet lbmethod=bytraffic
</Proxy>
```

```
<Proxy "http://backend">
  ProxySet keepalive=0n
</Proxy>
```

```
ProxySet "balancer://foo" lbmethod=bytraffic
```

```
ProxySet "ajp://backend:7001" timeout=15
```

**Warning**

Keep in mind that the same parameter key can have a different meaning depending whether it is applied to a balancer or a worker, as shown by the two examples above regarding timeout.



## ProxySourceAddress Directive

<b>Description:</b>	Set local IP address for outgoing proxy connections
<b>Syntax:</b>	ProxySourceAddress <i>address</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in version 2.3.9 and later

This directive allows to set a specific local address to bind to when connecting to a backend server.



## ProxyStatus Directive

<b>Description:</b>	Show Proxy LoadBalancer status in mod_status
<b>Syntax:</b>	ProxyStatus Off On Full
<b>Default:</b>	ProxyStatus Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in version 2.2 and later

This directive determines whether or not proxy loadbalancer status data is displayed via the [mod\\_status](#) server-status page.

### Note

**Full** is synonymous with **On**



<b>Description:</b>	Network timeout for proxied requests
<b>Syntax:</b>	ProxyTimeout <i>seconds</i>
<b>Default:</b>	Value of <a href="#">Timeout</a>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive allows a user to specify a timeout on proxy requests. This is useful when you have a slow/buggy appserver which hangs, and you would rather just return a timeout and fail gracefully instead of waiting however long it takes the server to return.



<b>Description:</b>	Information provided in the <code>Via</code> HTTP response header for proxied requests
<b>Syntax:</b>	<code>ProxyVia On Off Full Block</code>
<b>Default:</b>	<code>ProxyVia Off</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_proxy</code>

This directive controls the use of the `Via`: HTTP header by the proxy. Its intended use is to control the flow of proxy requests along a chain of proxy servers. See [RFC 2616](#) (HTTP/1.1), section 14.45 for an explanation of `Via`: header lines.

- If set to `Off`, which is the default, no special processing is performed. If a request or reply contains a `Via`: header, it is passed through unchanged.
- If set to `On`, each request and reply will get a `Via`: header line added for the current host.
- If set to `Full`, each generated `Via`: header line will additionally have the Apache httpd server version shown as a `Via`: comment field.
- If set to `Block`, every proxy request will have all its `Via`: header lines removed. No new `Via`: header will be generated.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_ajp`

<b>Description:</b>	AJP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_ajp_module</code>
<b>Source File:</b>	<code>mod_proxy_ajp.c</code>
<b>Compatibility:</b>	Available in version 2.1 and later

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides support for the Apache JServ Protocol version 1.3 (hereafter *AJP13*).

Thus, in order to get the ability of handling AJP13 protocol, [mod\\_proxy](#) and [mod\\_proxy\\_ajp](#) have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[Environment Variable documentation](#)



This module is used to reverse proxy to a backend application server (e.g. Apache Tomcat) using the AJP13 protocol. The usage is similar to an HTTP reverse proxy, but uses the `ajp://` prefix:

### Simple Reverse Proxy

```
ProxyPass "/app" "ajp://backend.example.com:8009/app"
```

Balancers may also be used:

### Balancer Reverse Proxy

```
<Proxy "balancer://cluster">
  BalancerMember "ajp://app1.example.com:8009" loadfactor=1
  BalancerMember "ajp://app2.example.com:8009" loadfactor=2
  ProxySet lbmethod=bytraffic
</Proxy>
ProxyPass "/app" "balancer://cluster/app"
```

Note that usually no [ProxyPassReverse](#) directive is necessary. The AJP request includes the original host header given to the proxy, and the application server can be expected to generate self-referential headers relative to this host, so no rewriting is necessary.

The main exception is when the URL path on the proxy differs from that on the backend. In this case, a redirect header can be rewritten relative to the original host URL (not the backend `ajp://` URL), for example:

### Rewriting Proxied Path

```
ProxyPass "/apps/foo" "ajp://backend.example.com:8009/foo"
ProxyPassReverse "/apps/foo" "http://www.example.com/foo"
```

However, it is usually better to deploy the application on the backend server at the same path as the proxy rather than to take

this approach.



## Environment Variables

Environment variables whose names have the prefix `AJP_` are forwarded to the origin server as AJP request attributes (with the `AJP_` prefix removed from the name of the key).



## Operation of the protocol

The AJP13 protocol is packet-oriented. A binary format was presumably chosen over the more readable plain text for reasons of performance. The web server communicates with the servlet container over TCP connections. To cut down on the expensive process of socket creation, the web server will attempt to maintain persistent TCP connections to the servlet container, and to reuse a connection for multiple request/response cycles.

Once a connection is assigned to a particular request, it will not be used for any others until the request-handling cycle has terminated. In other words, requests are not multiplexed over connections. This makes for much simpler code at either end of the connection, although it does cause more connections to be open at once.

Once the web server has opened a connection to the servlet container, the connection can be in one of the following states:

- Idle  
No request is being handled over this connection.
- Assigned  
The connection is handling a specific request.

Once a connection is assigned to handle a particular request, the basic request information (e.g. HTTP headers, etc) is sent over the connection in a highly condensed form (e.g. common strings are encoded as integers). Details of that format are below in Request Packet Structure. If there is a body to the request (`content-length > 0`), that is sent in a separate packet immediately after.

At this point, the servlet container is presumably ready to start processing the request. As it does so, it can send the following messages back to the web server:

- **SEND\_HEADERS**  
Send a set of headers back to the browser.
- **SEND\_BODY\_CHUNK**  
Send a chunk of body data back to the browser.
- **GET\_BODY\_CHUNK**  
Get further data from the request if it hasn't all been transferred yet. This is necessary because the packets have a fixed maximum size and arbitrary amounts of data can be included the body of a request (for uploaded files, for example). (Note: this is unrelated to HTTP chunked transfer).
- **END\_RESPONSE**  
Finish the request-handling cycle.

Each message is accompanied by a differently formatted packet of data. See Response Packet Structures below for details.



There is a bit of an XDR heritage to this protocol, but it differs in lots of ways (no 4 byte alignment, for example).

AJP13 uses network byte order for all data types.

There are four data types in the protocol: bytes, booleans, integers and strings.

### **Byte**

A single byte.

### **Boolean**

A single byte, `1 = true`, `0 = false`. Using other non-zero values as true (i.e. C-style) may work in some places, but it won't in others.

### **Integer**

A number in the range of `0` to `2^16` (32768). Stored in 2 bytes with the high-order byte first.

### **String**

A variable-sized string (length bounded by `2^16`). Encoded with the length packed into two bytes first, followed by the string (including the terminating `\0`). Note that the encoded length does **not** include the trailing `\0` -- it is like `strlen`. This is a touch confusing on the Java side, which is littered with odd autoincrement statements to skip over these terminators. I believe the reason this was done was to allow the C code to be extra efficient when reading strings which the servlet container is sending back -- with the terminating `\0` character, the C code can pass around references into a single buffer, without copying. If the `\0` was missing, the C code would have to copy things out in order to get its notion of a string.

## Packet Size

According to much of the code, the max packet size is  $8 * 1024$  bytes (8K). The actual length of the packet is encoded in the header.

## Packet Headers

Packets sent from the server to the container begin with 0x1234. Packets sent from the container to the server begin with AB (that's the ASCII code for A followed by the ASCII code for B). After those first two bytes, there is an integer (encoded as above) with the length of the payload. Although this might suggest that the maximum payload could be as large as  $2^{16}$ , in fact, the code sets the maximum to be 8K.

<b>Packet Format (Server-&gt;Container)</b>					
Byte	0	1	2	3	4...(n+3)
Contents	0x12	0x34	Data	Length (n)	Data

<b>Packet Format (Container-&gt;Server)</b>					
Byte	0	1	2	3	4...(n+3)
Contents	A	B	Data	Length (n)	Data

For most packets, the first byte of the payload encodes the type of message. The exception is for request body packets sent from the server to the container -- they are sent with a standard packet header ( 0x1234 and then length of the packet), but without any prefix code after that.

The web server can send the following messages to the servlet container:

Code	Type of Packet	Meaning
------	----------------	---------

2	Forward Request	Begin the request-processing cycle with the following data
7	Shutdown	The web server asks the container to shut itself down.
8	Ping	The web server asks the container to take control (secure login phase).
10	CPing	The web server asks the container to respond quickly with a CPong.
none	Data	Size (2 bytes) and corresponding body data.

To ensure some basic security, the container will only actually do the Shutdown if the request comes from the same machine on which it's hosted.

The first Data packet is send immediately after the Forward Request by the web server.

The servlet container can send the following types of messages to the webserver:

Code	Type of Packet	Meaning
3	Send Body Chunk	Send a chunk of the body from the servlet container to the web server (and presumably, onto the browser).
4	Send Headers	Send the response headers from the servlet container to the web server (and presumably, onto the browser).
5	End Response	Marks the end of the response (and thus the request-handling cycle).
6	Get Body Chunk	Get further data from the request if it hasn't all been transferred yet.
9	CPong	The reply to a CPing request

Reply

Each of the above messages has a different internal structure, detailed below.



For messages from the server to the container of type *Forward Request*:

```
AJP13_FORWARD_REQUEST :=
    prefix_code      (byte) 0x02 = JK_AJP13_FORWARD_REQUEST
    method           (byte)
    protocol         (string)
    req_uri          (string)
    remote_addr     (string)
    remote_host     (string)
    server_name     (string)
    server_port     (integer)
    is_ssl          (boolean)
    num_headers     (integer)
    request_headers *(req_header_name req_header_value)
    attributes      *(attribut_name attribute_value)
    request_terminator (byte) 0xFF
```

The `request_headers` have the following structure:

```
req_header_name :=
    sc_req_header_name | (string) [see below for how this is par

sc_req_header_name := 0xA0xx (integer)

req_header_value := (string)
```

The `attributes` are optional and have the following structure:

```
attribute_name := sc_a_name | (sc_a_req_attribute string)

attribute_value := (string)
```

Not that the all-important header is `content-length`, because it determines whether or not the container looks for another packet immediately.

## Detailed description of the elements of Forward Request

## Request prefix

For all requests, this will be 2. See above for details on other Prefix codes.

## Method

The HTTP method, encoded as a single byte:

Command Name	Code
OPTIONS	1
GET	2
HEAD	3
POST	4
PUT	5
DELETE	6
TRACE	7
PROPFIND	8
PROPPATCH	9
MKCOL	10
COPY	11
MOVE	12
LOCK	13
UNLOCK	14
ACL	15
REPORT	16
VERSION-CONTROL	17
CHECKIN	18
CHECKOUT	19
UNCHECKOUT	20
SEARCH	21
MKWORKSPACE	22

UPDATE	23
LABEL	24
MERGE	25
BASELINE_CONTROL	26
MKACTIVITY	27

Later version of ajp13, will transport additional methods, even if they are not in this list.

**protocol, req\_uri, remote\_addr, remote\_host, server\_name, server\_port, is\_ssl**

These are all fairly self-explanatory. Each of these is required, and will be sent for every request.

**Headers**

The structure of request\_headers is the following: First, the number of headers num\_headers is encoded. Then, a series of header name req\_header\_name / value req\_header\_value pairs follows. Common header names are encoded as integers, to save space. If the header name is not in the list of basic headers, it is encoded normally (as a string, with prefixed length). The list of common headers sc\_req\_header\_name and their codes is as follows (all are case-sensitive):

Name	Code value	Code name
accept	0xA001	SC_REQ_ACCEPT
accept-charset	0xA002	SC_REQ_ACCEPT_CHARSET
accept-encoding	0xA003	SC_REQ_ACCEPT_ENCODING
accept-language	0xA004	SC_REQ_ACCEPT_LANGUAGE
authorization	0xA005	SC_REQ_AUTHORIZATION
connection	0xA006	SC_REQ_CONNECTION

content-type	0xA007	SC_REQ_CONTENT_TYPE
content-length	0xA008	SC_REQ_CONTENT_LENGTH
cookie	0xA009	SC_REQ_COOKIE
cookie2	0xA00A	SC_REQ_COOKIE2
host	0xA00B	SC_REQ_HOST
pragma	0xA00C	SC_REQ_PRAGMA
referer	0xA00D	SC_REQ_REFERER
user-agent	0xA00E	SC_REQ_USER_AGENT

The Java code that reads this grabs the first two-byte integer and if it sees an '0xA0' in the most significant byte, it uses the integer in the second byte as an index into an array of header names. If the first byte is not 0xA0, it assumes that the two-byte integer is the length of a string, which is then read in.

This works on the assumption that no header names will have length greater than 0x9FFF (==0xA000 - 1), which is perfectly reasonable, though somewhat arbitrary.

### **Note:**

The content-length header is extremely important. If it is present and non-zero, the container assumes that the request has a body (a POST request, for example), and immediately reads a separate packet off the input stream to get that body.

## **Attributes**

The attributes prefixed with a ? (e.g. ?context) are all optional. For each, there is a single byte code to indicate the type of attribute, and then its value (string or integer). They can be sent in any order (though the C code always sends them in the order listed below). A special terminating code is sent to signal the end of the list of optional attributes. The list of byte codes is:

Information	Code Value	Type Of Value	Note
?context	0x01	-	Not currently implemented
? servlet_path	0x02	-	Not currently implemented
? remote_user	0x03	String	
?auth_type	0x04	String	
? query_string	0x05	String	
?jvm_route	0x06	String	
?ssl_cert	0x07	String	
?ssl_cipher	0x08	String	
? ssl_session	0x09	String	
? req_attribute	0x0A	String	Name (the name of the attribute follows)
? ssl_key_size	0x0B	Integer	
are_done	0xFF	-	request_terminator

The context and servlet\_path are not currently set by the C code, and most of the Java code completely ignores whatever is sent over for those fields (and some of it will actually break if a string is sent along after one of those codes). I don't know if this is a bug or an unimplemented feature or just vestigial code, but it's missing from both sides of the connection.

The remote\_user and auth\_type presumably refer to HTTP-level authentication, and communicate the remote user's username and the type of authentication used to establish their identity (e.g. Basic, Digest).

The `query_string`, `ssl_cert`, `ssl_cipher`, and `ssl_session` refer to the corresponding pieces of HTTP and HTTPS.

The `jvm_route`, is used to support sticky sessions -- associating a user's session with a particular Tomcat instance in the presence of multiple, load-balancing servers.

Beyond this list of basic attributes, any number of other attributes can be sent via the `req_attribute` code `0x0A`. A pair of strings to represent the attribute name and value are sent immediately after each instance of that code. Environment values are passed in via this method.

Finally, after all the attributes have been sent, the attribute terminator, `0xFF`, is sent. This signals both the end of the list of attributes and also then end of the Request Packet.



for messages which the container can send back to the server.

```
AJP13_SEND_BODY_CHUNK :=
  prefix_code    3
  chunk_length  (integer)
  chunk          *(byte)
  chunk_terminator (byte) 0x00

AJP13_SEND_HEADERS :=
  prefix_code      4
  http_status_code (integer)
  http_status_msg  (string)
  num_headers      (integer)
  response_headers *(res_header_name header_value)

res_header_name :=
  sc_res_header_name | (string) [see below for how this is pa

sc_res_header_name := 0xA0 (byte)

header_value := (string)

AJP13_END_RESPONSE :=
  prefix_code    5
  reuse          (boolean)

AJP13_GET_BODY_CHUNK :=
  prefix_code      6
  requested_length (integer)
```

## Details:

### Send Body Chunk

The chunk is basically binary data, and is sent directly back to the browser.

### Send Headers

The status code and message are the usual HTTP things (e.g. 200 and OK). The response header names are encoded the same

way the request header names are. See `header_encoding` above for details about how the codes are distinguished from the strings. The codes for common headers are:

Name	Code value
Content-Type	0xA001
Content-Language	0xA002
Content-Length	0xA003
Date	0xA004
Last-Modified	0xA005
Location	0xA006
Set-Cookie	0xA007
Set-Cookie2	0xA008
Servlet-Engine	0xA009
Status	0xA00A
WWW-Authenticate	0xA00B

After the code or the string header name, the header value is immediately encoded.

## End Response

Signals the end of this request-handling cycle. If the `reuse` flag is true (anything other than 0 in the actual C code), this TCP connection can now be used to handle new incoming requests. If `reuse` is false (`==0`), the connection should be closed.

## Get Body Chunk

The container asks for more data from the request (If the body was too large to fit in the first packet sent over or when the request is chunked). The server will send a body packet back with an amount of data which is the minimum of the `request_length`,

the maximum send body size (8186 (8 Kbytes - 6)), and the number of bytes actually left to send from the request body. If there is no more data in the body (i.e. the servlet container is trying to read past the end of the body), the server will send back an *empty* packet, which is a body packet with a payload length of 0. (0x12, 0x34, 0x00, 0x00)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_balancer`

<b>Description:</b>	<code>mod_proxy</code> extension for load balancing
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_balancer_module</code>
<b>Source File:</b>	<code>mod_proxy_balancer.c</code>
<b>Compatibility:</b>	Available in version 2.1 and later

## Summary

This module *requires* the service of `mod_proxy` and it provides load balancing for all the supported protocols. The most important ones are:

- HTTP, using `mod_proxy http`
- FTP, using `mod_proxy ftp`
- AJP13, using `mod_proxy ajp`
- WebSocket, using `mod_proxy wstunnel`

The Load balancing scheduler algorithm is not provided by this module but from other ones such as:

- `mod_lbmethod byrequests`
- `mod_lbmethod bytraffic`
- `mod_lbmethod bybusyness`
- `mod_lbmethod heartbeat`

Thus, in order to get the ability of load balancing, `mod_proxy`, `mod_proxy_balancer` and at least one of load balancing scheduler algorithm modules have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open

proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[BalancerMember](#)

[BalancerGrowth](#)

[BalancerPersist](#)

[BalancerInherit](#)



---

At present, there are 3 load balancer scheduler algorithms available for use: Request Counting, Weighted Traffic Counting and Pending Request Counting. These are controlled via the `lbmethod` value of the Balancer definition. See the [ProxyPass](#) directive for more information, especially regarding how to configure the Balancer and BalancerMembers.



The balancer supports stickiness. When a request is proxied to some back-end, then all following requests from the same user should be proxied to the same back-end. Many load balancers implement this feature via a table that maps client IP addresses to back-ends. This approach is transparent to clients and back-ends, but suffers from some problems: unequal load distribution if clients are themselves hidden behind proxies, stickiness errors when a client uses a dynamic IP address that changes during a session and loss of stickiness, if the mapping table overflows.

The module [mod\\_proxy\\_balancer](#) implements stickiness on top of two alternative means: cookies and URL encoding. Providing the cookie can be either done by the back-end or by the Apache web server itself. The URL encoding is usually done on the back-end.



## Example of balancer configuration

Before we dive into the technical details, here's an example of how you might use [mod\\_proxy\\_balancer](#) to provide load balancing between two back-end servers:

```
<Proxy "balancer://mycluster">
    BalancerMember "http://192.168.1.50:80"
    BalancerMember "http://192.168.1.51:80"
</Proxy>
ProxyPass "/test" "balancer://mycluster"
ProxyPassReverse "/test" "balancer://mycluster"
```

Another example of how to provide load balancing with stickiness using [mod\\_headers](#), even if the back-end server does not set a suitable session cookie:

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_VHOST}"
<Proxy "balancer://mycluster">
    BalancerMember "http://192.168.1.50:80"
    BalancerMember "http://192.168.1.51:80"
    ProxySet sticky-session=ROUTEID
</Proxy>
ProxyPass "/test" "balancer://mycluster"
ProxyPassReverse "/test" "balancer://mycluster"
```



At present there are 6 environment variables exported:

***BALANCER\_SESSION\_STICKY***

This is assigned the *stickysession* value used for the current request. It is the name of the cookie or request parameter used for sticky sessions

***BALANCER\_SESSION\_ROUTE***

This is assigned the *route* parsed from the current request.

***BALANCER\_NAME***

This is assigned the name of the balancer used for the current request. The value is something like `balancer://foo`.

***BALANCER\_WORKER\_NAME***

This is assigned the name of the worker used for the current request. The value is something like `http://hostA:1234`.

***BALANCER\_WORKER\_ROUTE***

This is assigned the *route* of the worker that will be used for the current request.

***BALANCER\_ROUTE\_CHANGED***

This is set to 1 if the session route does not match the worker route (`BALANCER_SESSION_ROUTE != BALANCER_WORKER_ROUTE`) or the session does not yet have an established route. This can be used to determine when/if the client needs to be sent an updated route when sticky sessions are used.



## Enabling Balancer Manager Support

This module *requires* the service of `mod_status`. Balancer manager enables dynamic update of balancer members. You can use balancer manager to change the balance factor of a particular member, or put it in the off line mode.

Thus, in order to get the ability of load balancer management, `mod_status` and `mod_proxy_balancer` have to be present in the server.

To enable load balancer management for browsers from the example.com domain add this code to your `httpd.conf` configuration file

```
<Location "/balancer-manager">  
    SetHandler balancer-manager  
    Require host example.com  
</Location>
```

You can now access load balancer manager by using a Web browser to access the page `http://your.server.name/balancer-manager`. Please note that only Balancers defined outside of `<Location ...>` containers can be dynamically controlled by the Manager.



## Cookie based stickiness

When using cookie based stickiness, you need to configure the name of the cookie that contains the information about which back-end to use. This is done via the *stickysession* attribute added to either [ProxyPass](#) or [ProxySet](#). The name of the cookie is case-sensitive. The balancer extracts the value of the cookie and looks for a member worker with *route* equal to that value. The *route* must also be set in either [ProxyPass](#) or [ProxySet](#). The cookie can either be set by the back-end, or as shown in the above [example](#) by the Apache web server itself.

Some back-ends use a slightly different form of stickiness cookie, for instance Apache Tomcat. Tomcat adds the name of the Tomcat instance to the end of its session id cookie, separated with a dot (.) from the session id. Thus if the Apache web server finds a dot in the value of the stickiness cookie, it only uses the part behind the dot to search for the route. In order to let Tomcat know about its instance name, you need to set the attribute `jvmRoute` inside the Tomcat configuration file `conf/server.xml` to the value of the *route* of the worker that connects to the respective Tomcat. The name of the session cookie used by Tomcat (and more generally by Java web applications based on servlets) is `JSESSIONID` (upper case) but can be configured to something else.

The second way of implementing stickiness is URL encoding. The web server searches for a query parameter in the URL of the request. The name of the parameter is specified again using *stickysession*. The value of the parameter is used to lookup a member worker with *route* equal to that value. Since it is not easy to extract and manipulate all URL links contained in responses, generally the work of adding the parameters to each link is done by the back-end generating the content. In some cases it might be feasible doing this via the web server using [mod\\_substitute](#) or

[mod\\_sed](#). This can have negative impact on performance though.

The Java standards implement URL encoding slightly different. They use a path info appended to the URL using a semicolon (;) as the separator and add the session id behind. As in the cookie case, Apache Tomcat can include the configured `jvmRoute` in this path info. To let Apache find this sort of path info, you need to set `scolonpathdelim` to `On` in [ProxyPass](#) or [ProxySet](#).

Finally you can support cookies and URL encoding at the same time, by configuring the name of the cookie and the name of the URL parameter separated by a vertical bar (|) as in the following example:

```
ProxyPass "/test" "balancer://mycluster" st:
<Proxy "balancer://mycluster">
    BalancerMember "http://192.168.1.50:80"
    BalancerMember "http://192.168.1.51:80"
</Proxy>
```

If the cookie and the request parameter both provide routing information for the same request, the information from the request parameter is used.



If you experience stickyness errors, e.g. users lose their application sessions and need to login again, you first want to check whether this is because the back-ends are sometimes unavailable or whether your configuration is wrong. To find out about possible stability problems with the back-ends, check your Apache error log for proxy error messages.

To verify your configuration, first check, whether the stickyness is based on a cookie or on URL encoding. Next step would be logging the appropriate data in the access log by using an enhanced [LogFormat](#). The following fields are useful:

**%{MYCOOKIE}C**

The value contained in the cookie with name MYCOOKIE. The name should be the same given in the *stickysession* attribute.

**%{Set-Cookie}o**

This logs any cookie set by the back-end. You can track, whether the back-end sets the session cookie you expect, and to which value it is set.

**%{BALANCER\_SESSION\_STICKY}e**

The name of the cookie or request parameter used to lookup the routing information.

**%{BALANCER\_SESSION\_ROUTE}e**

The route information found in the request.

**%{BALANCER\_WORKER\_ROUTE}e**

The route of the worker chosen.

**%{BALANCER\_ROUTE\_CHANGED}e**

Set to 1 if the route in the request is different from the route of the worker, i.e. the request couldn't be handled sticky.

Common reasons for loss of session are session timeouts, which are usually configurable on the back-end server.

The balancer also logs detailed information about handling stickyness to the error log, if the log level is set to debug or higher. This is an easy way to troubleshoot stickyness problems, but the log volume might be too high for production servers under high load.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_connect`

<b>Description:</b>	<code>mod_proxy</code> extension for CONNECT request handling
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_connect_module</code>
<b>Source File:</b>	<code>mod_proxy_connect.c</code>

## Summary

This module *requires* the service of `mod_proxy`. It provides support for the CONNECT HTTP method. This method is mainly used to tunnel SSL requests through proxy servers.

Thus, in order to get the ability of handling CONNECT requests, `mod_proxy` and `mod_proxy_connect` have to be present in the server.

CONNECT is also used when the server needs to send an HTTPS request through a forward proxy. In this case the server acts as a CONNECT client. This functionality is part of `mod_proxy` and `mod_proxy_connect` is not needed in this case.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)



## Request Notes

`mod_proxy_connect` creates the following request notes for logging using the `%{VARNAME}n` format in [LogFormat](#) or [ErrorLogFormat](#):

### **proxy-source-port**

The local port used for the connection to the backend server.



<b>Description:</b>	Ports that are allowed to CONNECT through the proxy
<b>Syntax:</b>	<code>AllowCONNECT port[-port] [port[-port]] ...</code>
<b>Default:</b>	<code>AllowCONNECT 443 563</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_proxy_connect</code>
<b>Compatibility:</b>	Moved from <code>mod_proxy</code> in Apache 2.3.5. Port ranges available since Apache 2.3.7.

The `AllowCONNECT` directive specifies a list of port numbers or ranges to which the proxy CONNECT method may connect. Today's browsers use this method when a `https` connection is requested and proxy tunneling over HTTP is in effect.

By default, only the default `https` port (443) and the default `snmp` port (563) are enabled. Use the `AllowCONNECT` directive to override this default and allow connections to the listed ports only.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_express`

<b>Description:</b>	Dynamic mass reverse proxy extension for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_express_module</code>
<b>Source File:</b>	<code>mod_proxy_express.c</code>

## Summary

This module creates dynamically configured mass reverse proxies, by mapping the `Host:` header of the HTTP request to a server name and backend URL stored in a DBM file. This allows for easy use of a huge number of reverse proxies with no configuration changes. It is much less feature-full than [mod\\_proxy\\_balancer](#), which also provides dynamic growth, but is intended to handle much, much larger numbers of backends. It is ideally suited as a front-end HTTP switch and for micro-services architectures.

This module *requires* the service of [mod\\_proxy](#).

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

## Limitations

- This module is not intended to replace the dynamic capability of [mod\\_proxy\\_balancer](#). Instead, it is intended to be mostly a lightweight and fast alternative to using [mod\\_rewrite](#) with `RewriteMap` and the `[P]` flag for mapped reverse proxying.
- It does not support regex or pattern matching at all.

- It emulates:

```
<VirtualHost *:80>
  ServerName front.end.server
  ProxyPass "/" "back.end.server:port"
  ProxyPassReverse "/" "back.end.server:port"
</VirtualHost>
```

That is, the entire URL is appended to the mapped backend URL. This is in keeping with the intent of being a simple but fast reverse proxy switch.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[BalancerMember](#)

[BalancerGrowth](#)

[BalancerPersist](#)

[BalancerInherit](#)



<b>Description:</b>	Pathname to DBM file.
<b>Syntax:</b>	ProxyExpressDBMFile <pathname>
<b>Default:</b>	None
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_express
<b>Compatibility:</b>	Available in Apache 2.3.13 and later

The `ProxyExpressDBMFile` directive points to the location of the Express map DBM file. This file serves to map the incoming server name, obtained from the Host: header, to a backend URL.

## Note

The file is constructed from a plain text file format using the [httxt2dbm](#) utility.

### ProxyExpress map file

```
##
##express-map.txt:
##

www1.example.com http://192.168.211.2:8080
www2.example.com http://192.168.211.12:8088
www3.example.com http://192.168.212.10
```

### Create DBM file

```
httxt2dbm -i express-map.txt -o emap
```

### Configuration

```
ProxyExpressEnable on
ProxyExpressDBMFile emap
```



<b>Description:</b>	DBM type of file.
<b>Syntax:</b>	ProxyExpressDBMFile <type>
<b>Default:</b>	"default"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_express
<b>Compatibility:</b>	Available in Apache 2.3.13 and later

The `ProxyExpressDBMType` directive controls the DBM type expected by the module. The default is the default DBM type created with [httxt2dbm](#).

Possible values are (not all may be available at run time):

Value	Description
db	Berkeley DB files
gdbm	GDBM files
ndbm	NDBM files
sdbm	SDBM files (always available)
default	default DBM type



## ProxyExpressEnable Directive

<b>Description:</b>	Enable the module functionality.
<b>Syntax:</b>	ProxyExpressEnable [on off]
<b>Default:</b>	off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_express
<b>Compatibility:</b>	Available in Apache 2.3.13 and later

The **ProxyExpressEnable** directive controls whether the module will be active.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_fcgi`

<b>Description:</b>	FastCGI support module for <code>mod_proxy</code>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_fcgi_module</code>
<b>Source File:</b>	<code>mod_proxy_fcgi.c</code>
<b>Compatibility:</b>	Available in version 2.3 and later

## Summary

This module *requires* the service of `mod_proxy`. It provides support for the [FastCGI](#) protocol.

Thus, in order to get the ability of handling the FastCGI protocol, `mod_proxy` and `mod_proxy_fcgi` have to be present in the server.

Unlike `mod_fcgid` and `mod_fastcgi`, `mod_proxy_fcgi` has no provision for starting the application process; `fcgistarter` is provided (on some platforms) for that purpose. Alternatively, external launching or process management may be available in the FastCGI application framework in use.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[fcgistarter](#)

[mod\\_proxy](#)

[mod\\_authnz\\_fcgi](#)



Remember, in order to make the following examples work, you have to enable [mod\\_proxy](#) and [mod\\_proxy\\_fcgi](#).

### Single application instance

```
ProxyPass "/myapp/" "fcgi://localhost:4000/"
```

[mod\\_proxy\\_fcgi](#) disables connection reuse by default, so after a request has been completed the connection will NOT be held open by that httpd child process and won't be reused. If the FastCGI application is able to handle concurrent connections from httpd, you can opt-in to connection reuse as shown in the following example:

### Single application instance, connection reuse (2.4.11 and later)

```
ProxyPass "/myapp/" "fcgi://localhost:4000/" enable reuse=on
```

The following example passes the request URI as a filesystem path for the PHP-FPM daemon to run. The request URL is implicitly added to the 2nd parameter. The hostname and port following `fcgi://` are where PHP-FPM is listening. Connection pooling/reuse is enabled.

### PHP-FPM

```
ProxyPassMatch "^/myapp/.*\.php(/.*)?$" "fcgi://localhost:9000/\u2002
```

The following example passes the request URI as a filesystem path for the PHP-FPM daemon to run. In this case, PHP-FPM is listening on a unix domain socket (UDS). Requires 2.4.9 or later. With this syntax, the hostname and optional port following `fcgi://` are ignored.

## PHP-FPM with UDS

```
ProxyPassMatch "^/(.*\.php(/.*)?)" "unix:/var/run/php5-fpm.sock"
```

The balanced gateway needs [mod\\_proxy\\_balancer](#) and at least one load balancer algorithm module, such as [mod\\_lbmethod\\_byrequests](#), in addition to the proxy modules listed above. [mod\\_lbmethod\\_byrequests](#) is the default, and will be used for this example configuration.

## Balanced gateway to multiple application instances

```
ProxyPass "/myapp/" "balancer://myappcluster/"  
<Proxy "balancer://myappcluster/">  
    BalancerMember "fcgi://localhost:4000"  
    BalancerMember "fcgi://localhost:4001"  
</Proxy>
```

You can also force a request to be handled as a reverse-proxy request, by creating a suitable Handler pass-through. The example configuration below will pass all requests for PHP scripts to the specified FastCGI server using reverse proxy. This feature is available in Apache HTTP Server 2.4.10 and later. For performance reasons, you will want to define a [worker](#) representing the same fcgi:// backend. The benefit of this form is that it allows the normal mapping of URI to filename to occur in the server, and the local filesystem result is passed to the backend. When FastCGI is configured this way, the server can calculate the most accurate PATH\_INFO.

## Proxy via Handler

```
<FilesMatch "\.php$">  
    # Note: The only part that varies is /path/to/app.sock  
    SetHandler "proxy:unix:/path/to/app.sock|fcgi://localhost/"  
</FilesMatch>
```

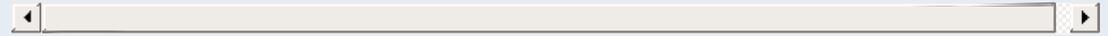
```
# Define a matching worker.  
# The part that is matched to the SetHandler is the part that
```

```
# follows the pipe. If you need to distinguish, "localhost;" can  
# be anything unique.
```

```
<Proxy "fcgi://localhost/" enablereuse=on max=10>  
</Proxy>
```

```
<FilesMatch ...>  
    SetHandler "proxy:fcgi://localhost:9000"  
</FilesMatch>
```

```
<FilesMatch ...>  
    SetHandler "proxy:balancer://myappcluster/"  
</FilesMatch>
```



In addition to the configuration directives that control the behaviour of `mod_proxy`, there are a number of *environment variables* that control the FCGI protocol provider:

### **proxy-fcgi-pathinfo**

When configured via `ProxyPass` or `ProxyPassMatch`, `mod_proxy_fcgi` will not set the `PATH_INFO` environment variable. This allows the backend FCGI server to correctly determine `SCRIPT_NAME` and `Script-URI` and be compliant with RFC 3875 section 3.3. If instead you need `mod_proxy_fcgi` to generate a "best guess" for `PATH_INFO`, set this env-var. This is a workaround for a bug in some FCGI implementations. This variable can be set to multiple values to tweak at how the best guess is chosen (In 2.4.11 and later only):

#### **first-dot**

`PATH_INFO` is split from the slash following the *first* "." in the URL.

#### **last-dot**

`PATH_INFO` is split from the slash following the *last* "." in the URL.

#### **full**

`PATH_INFO` is calculated by an attempt to map the URL to the local filesystem.

#### **unescape**

`PATH_INFO` is the path component of the URL, unescaped / decoded.

#### **any other value**

`PATH_INFO` is the same as the path component of the URL. Originally, this was the only proxy-fcgi-pathinfo option.



## ProxyFCGIBackendType Directive

<b>Description:</b>	Specify the type of backend FastCGI application
<b>Syntax:</b>	ProxyFCGIBackendType FPM GENERIC
<b>Default:</b>	ProxyFCGIBackendType FPM
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_fcgi
<b>Compatibility:</b>	Available in version 2.4.26 and later

This directive allows the type of backend FastCGI application to be specified. Some FastCGI servers, such as PHP-FPM, use historical quirks of environment variables to identify the type of proxy server being used. Set this directive to "GENERIC" if your non PHP-FPM application has trouble interpreting environment variables such as SCRIPT\_FILENAME or PATH\_TRANSLATED as set by the server.

One example of values that change based on the setting of this directive is SCRIPT\_FILENAME. When using [mod\\_proxy\\_fcgi](#) historically, SCRIPT\_FILENAME was prefixed with the string "proxy:fcgi://". This variable is what some generic FastCGI applications would read as their script input, but PHP-FPM would strip the prefix then remember it was talking to Apache. In 2.4.21 through 2.4.25, this prefix was automatically stripped by the server, breaking the ability of PHP-FPM to detect and interoperate with Apache in some scenarios.



<b>Description:</b>	Allow variables sent to FastCGI servers to be fixed up
<b>Syntax:</b>	<code>ProxyFCGISetEnvIf <i>conditional-expression</i> [!]<i>environment-variable-name</i> [<i>value-expression</i>]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_fcgi
<b>Compatibility:</b>	Available in version 2.4.26 and later

Just before passing a request to the configured FastCGI server, the core of the web server sets a number of environment variables based on details of the current request. FastCGI programs often uses these environment variables as inputs that determine what underlying scripts they will process, or what output they directly produce.

Examples of noteworthy environment variables are:

- SCRIPT\_NAME
- SCRIPT\_FILENAME
- REQUEST\_URI
- PATH\_INFO
- PATH\_TRANSLATED

This directive allows the environment variables above, or any others of interest, to be overridden. This directive is evaluated after the initial values for these variables are set, so they can be used as input into both the condition expressions and value expressions.

Parameter syntax:

**conditional-expression**

Specifies an expression that controls whether the environment variable that follows will be modified. For information on the expression syntax, see the examples that follow or the full specification at the [ap\\_expr](#) documentation.

### **environment-variable-name**

Specifies the CGI environment variable to change, such as PATH\_INFO. If preceded by an exclamation point, the variable will be unset.

### **value-expression**

Specifies the replacement value for the preceding environment variable. Backreferences, such as "\$1", can be included from regular expression captures in *conditional-expression*. If omitted, the variable is set (or overridden) to an empty string — but see the Note below.

```
# A basic, unconditional override
ProxyFCGISetEnvIf "true" PATH_INFO "/example"

# Use an environment variable in the value
ProxyFCGISetEnvIf "true" PATH_INFO "%{reqenv:SCRIPT_NAME}"

# Use captures in the conditions and backreferences in the repl:
ProxyFCGISetEnvIf "reqenv('PATH_TRANSLATED') =~ m|(/.*prefix)(\c
```

### **Note: Unset vs. Empty**

The following will unset VARIABLE, preventing it from being sent to the FastCGI server:

```
ProxyFCGISetEnvIf true !VARIABLE
```

Whereas the following will erase any existing *value* of VARIABLE (by setting it to the empty string), but the empty VARIABLE will still be sent to the server:

```
ProxyFCGISetEnvIf true VARIABLE
```

The CGI/1.1 specification [does not distinguish](#) between a variable with an empty value and a variable that does not exist. However, many CGI and FastCGI implementations distinguish (or allow scripts to distinguish) between the two. The choice of which to use is dependent upon your implementation and your reason for modifying the variable.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_fdpass`

<b>Description:</b>	fdpass external process support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	proxy_fdpass_module
<b>Source File:</b>	mod_proxy_fdpass.c
<b>Compatibility:</b>	Available for unix in version 2.3 and later

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides support for the passing the socket of the client to another process.

`mod_proxy_fdpass` uses the ability of AF\_UNIX domain sockets to [pass an open file descriptor](#) to allow another process to finish handling a request.

The module has a `proxy_fdpass_flusher` provider interface, which allows another module to optionally send the response headers, or even the start of the response body. The default `flush` provider disables keep-alive, and sends the response headers, letting the external process just send a response body.

In order to use another provider, you have to set the `flusher` parameter in the [ProxyPass](#) directive.

At this time the only data passed to the external process is the client socket. To receive a client socket, call `recvfrom` with an allocated [struct cmsghdr](#). Future versions of this module may include more data after the client socket, but this is not implemented at this time.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_ftp`

<b>Description:</b>	FTP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_ftp_module</code>
<b>Source File:</b>	<code>mod_proxy_ftp.c</code>

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides support for the proxying FTP sites. Note that FTP support is currently limited to the GET method.

Thus, in order to get the ability of handling FTP proxy requests, [mod\\_proxy](#) and [mod\\_proxy\\_ftp](#) have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

mod\_proxy



You probably don't have that particular file type defined as `application/octet-stream` in your proxy's `mime.types` configuration file. A useful line can be

```
application/octet-stream  bin dms lha lzh exe class tgz taz
```

Alternatively you may prefer to default everything to binary:

```
ForceType application/octet-stream
```



---

In the rare situation where you must download a specific file using the FTP ASCII transfer method (while the default transfer is in binary mode), you can override [mod\\_proxy](#)'s default by suffixing the request with `;type=a` to force an ASCII transfer. (FTP Directory listings are always executed in ASCII mode, however.)



## How can we do HTTP upload?

Currently, only GET is supported for FTP in mod\_proxy. You can of course use HTTP upload (POST or PUT) through an Apache proxy.



## directory?

An FTP URI is interpreted relative to the home directory of the user who is logging in. Alas, to reach higher directory levels you cannot use `../`, as the dots are interpreted by the browser and not actually sent to the FTP server. To address this problem, the so called *Squid %2f hack* was implemented in the Apache FTP proxy; it is a solution which is also used by other popular proxy servers like the [Squid Proxy Cache](#). By prepending `/%2f` to the path of your request, you can make such a proxy change the FTP starting directory to `/` (instead of the home directory). For example, to retrieve the file `/etc/motd`, you would use the URL:

```
ftp://user@host/%2f/etc/motd
```



## How can I hide the FTP cleartext password in my browser's URL line?

To log in to an FTP server by username and password, Apache uses different strategies. In absence of a user name and password in the URL altogether, Apache sends an anonymous login to the FTP server, *i.e.*,

```
user: anonymous  
password: apache_proxy@
```

This works for all popular FTP servers which are configured for anonymous access.

For a personal login with a specific username, you can embed the user name into the URL, like in:

```
ftp://username@host/myfile
```

If the FTP server asks for a password when given this username (which it should), then Apache will reply with a 401 (Authorization required) response, which causes the Browser to pop up the username/password dialog. Upon entering the password, the connection attempt is retried, and if successful, the requested resource is presented. The advantage of this procedure is that your browser does not display the password in cleartext (which it would if you had used

```
ftp://username:password@host/myfile
```

in the first place).

### Note

The password which is transmitted in such a way is not encrypted on its way. It travels between your browser and the

Apache proxy server in a base64-encoded cleartext string, and between the Apache proxy and the FTP server as plaintext. You should therefore think twice before accessing your FTP server via HTTP (or before accessing your personal files via FTP at all!) When using insecure channels, an eavesdropper might intercept your password on its way.



## downloaded?

In order to allow both browsing the directories on an FTP server and downloading files, Apache looks at the request URL. If it looks like a directory, or contains wildcard characters ("\*?{~"), then it guesses that a listing is wanted instead of a download.

You can disable the special handling of names with wildcard characters. See the `ProxyFtpListOnWildcard` directive.



## ProxyFtpDirCharset Directive

<b>Description:</b>	Define the character set for proxied FTP listings
<b>Syntax:</b>	ProxyFtpDirCharset <i>character set</i>
<b>Default:</b>	ProxyFtpDirCharset ISO-8859-1
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_ftp
<b>Compatibility:</b>	Available in Apache 2.2.7 and later. Moved from <a href="#">mod_proxy</a> in Apache 2.3.5.

The `ProxyFtpDirCharset` directive defines the character set to be set for FTP directory listings in HTML generated by [mod\\_proxy\\_ftp](#).



## ProxyFtpEscapewildcards Directive

<b>Description:</b>	Whether wildcards in requested filenames are escaped when sent to the FTP server
<b>Syntax:</b>	ProxyFtpEscapewildcards [on off]
<b>Default:</b>	on
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_ftp
<b>Compatibility:</b>	Available in Apache 2.3.3 and later

The `ProxyFtpEscapewildcards` directive controls whether wildcard characters ("\*?[{~") in requested filenames are escaped with backslash before sending them to the FTP server. That is the default behavior, but many FTP servers don't know about the escaping and try to serve the literal filenames they were sent, including the backslashes in the names.

Set to "off" to allow downloading files with wildcards in their names from FTP servers that don't understand wildcard escaping.



<b>Description:</b>	Whether wildcards in requested filenames trigger a file listing
<b>Syntax:</b>	ProxyFtpListOnWildcard [on off]
<b>Default:</b>	on
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_ftp
<b>Compatibility:</b>	Available in Apache 2.3.3 and later

The `ProxyFtpListOnWildcard` directive controls whether wildcard characters ("\*?{~") in requested filenames cause `mod_proxy_ftp` to return a listing of files instead of downloading a file. By default (value on), they do. Set to "off" to allow downloading files even if they have wildcard characters in their names.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_hcheck`

<b>Description:</b>	Dynamic health check of Balancer members (workers) for <code>mod_proxy</code>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_hcheck_module</code>
<b>Source File:</b>	<code>mod_proxy_hcheck.c</code>
<b>Compatibility:</b>	Available in Apache 2.4.21 and later

## Summary

This module provides for dynamic health checking of balancer members (workers). This can be enabled on a worker-by-worker basis. The health check is done independently of the actual reverse proxy requests.

This module *requires* the service of `mod_watchdog`.

## Parameters

The health check mechanism is enabled via the use of additional BalancerMember parameters, which are configured in the standard way via `ProxyPass`:

A new BalancerMember status state (flag) is defined via this module: "C". When the worker is taken offline due to failures as determined by the health check module, this flag is set, and can be seen (and modified) via the `balancer-manager`.

### Parameter Default Description

<code>hcmethod</code>	None	No dynamic health check performed. Choices are:
-----------------------	------	---

Method	Description	Note
None	No dynamic health	

checking done

TCP Check that a socket to the backend can be created: e.g. "are you up"

OPTIONS Send an HTTP OPTIONS request to the backend \*

HEAD Send an HTTP HEAD request to the backend \*

GET Send an HTTP GET request to the backend \*

\*: Unless `hcexpr` is used, a 2xx or 3xx HTTP status will be interpreted as *passing* the health check

<code>hcpasses</code>	1	Number of successful health check tests before worker is re-enabled
<code>hcfails</code>	1	Number of failed health check tests before worker is disabled
<code>hcinterval</code>	30	Period of health checks in seconds (e.g. performed every 30 seconds)
<code>hcuri</code>		Additional URI to be appended to the worker URL for the health check.
<code>hctemplate</code>		Name of template, created via <code>ProxyHCTemplate</code> to use for setting health check parameters for this worker
<code>hcexpr</code>		Name of expression, created via <code>ProxyHCExpr</code> , used to check response headers for health. <i>If not used, 2xx thru 3xx status codes imply success</i>



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)



The following example shows how one might configured health checking for various backend servers:

```
ProxyHCExpr ok234 {%{REQUEST_STATUS} =~ /^[234]
ProxyHCExpr gdown {%{REQUEST_STATUS} =~ /^[56]
ProxyHCExpr in_maint {hc('body') !~ /Under r

<Proxy balancer://foo>
  BalancerMember http://www.example.com/ hc
  BalancerMember http://www2.example.com/ h
  BalancerMember http://www3.example.com/ hc
  BalancerMember http://www4.example.com/
</Proxy>

ProxyPass "/" "balancer://foo"
ProxyPassReverse "/" "balancer://foo"
```

In this scenario, `http://www.example.com/` is health checked by sending a `GET /status.php` request to that server and seeing that the returned page does not include the string *Under maintenance*. If it does, that server is put in health-check fail mode, and disabled. This dynamic check is performed every 30 seconds, which is the default.

`http://www2.example.com/` is checked by sending a simple `HEAD` request every 10 seconds and making sure that the response status is 2xx, 3xx or 4xx.

`http://www3.example.com/` is checked every 5 seconds by simply ensuring that the socket to that server is up. If the backend is marked as "down" and it passes 2 health check, it will be re-enabled and added back into the load balancer. It takes 3 back-to-back health check failures to disable the server and move it out of rotation. Finally, `http://www4.example.com/` is not

dynamically checked at all.



<b>Description:</b>	Creates a named condition expression to use to determine health of the backend based on its response.
<b>Syntax:</b>	ProxyHCEpr name {ap_expr expression}
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_hcheck

The **ProxyHCEpr** directive allows for creating a named condition expression that checks the response headers of the backend server to determine its health. This named condition can then be assigned to balancer members via the `hcexpr` parameter

**ProxyHCEpr: Allow for 2xx/3xx/4xx as passing**

```
ProxyHCEpr ok234 {%{REQUEST_STATUS} =~ /^[234]/}
ProxyPass "/apps" "http://backend.example.com/" hcexpr=ok234
```

The [expression](#) can use curly-parens ("{}") as quoting delimiters in addition to normal quotes.

If using a health check method (eg: GET) which results in a response body, that body itself can be checked via `ap_expr` using the `hc()` expression function, which is unique to this module.

In the following example, we send the backend a GET request and if the response body contains the phrase *Under maintenance*, we want to disable the backend.

**ProxyHCEpr: Checking response body**

```
ProxyHCEpr in_maint {hc('body') !~ /Under maintenance/}
ProxyPass "/apps" "http://backend.example.com/" hcexpr=in_maint
```

*NOTE:* Since response body can quite large, it is best if used against specific status pages.



## ProxyHCTemplate Directive

<b>Description:</b>	Creates a named template for setting various health check parameters
<b>Syntax:</b>	ProxyHCTemplate name parameter=setting <...>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_hcheck

The `ProxyHCTemplate` directive allows for creating a named set (template) of health check parameters that can then be assigned to balancer members via the `hctemplate` parameter

### ProxyHCTemplate

```
ProxyHCTemplate tcp5 hcmethod=tcp hcinterval=5  
ProxyPass "/apps" "http://backend.example.com/" hctemplate=1
```



**Description:** Sets the total server-wide size of the threadpool used for the health check workers.

**Syntax:** ProxyHCTPsize <size>

**Context:** server config

**Status:** Extension

**Module:** mod\_proxy\_hcheck

If Apache httpd and APR are built with thread support, the health check module will offload the work of the actual checking to a threadpool associated with the Watchdog process, allowing for parallel checks. The `ProxyHCTPsize` directive determines the size of this threadpool. If set to 0, no threadpool is used at all, resulting in serialized health checks. The default size is 16.

## ProxyHCTPsize

ProxyHCTPsize 32

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_html`

<b>Description:</b>	Rewrite HTML links in to ensure they are addressable from Clients' networks in a proxy context.
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>proxy_html_module</code>
<b>Source File:</b>	<code>mod_proxy_html.c</code>
<b>Compatibility:</b>	Version 2.4 and later. Available as a third-party module for earlier 2.x versions

## Summary

This module provides an output filter to rewrite HTML links in a proxy situation, to ensure that links work for users outside the proxy. It serves the same purpose as Apache's `ProxyPassReverse` directive does for HTTP headers, and is an essential component of a reverse proxy.

For example, if a company has an application server at `appserver.example.com` that is only visible from within the company's internal network, and a public webserver `www.example.com`, they may wish to provide a gateway to the application server at `http://www.example.com/appserver/`. When the application server links to itself, those links need to be rewritten to work through the gateway. `mod_proxy_html` serves to rewrite `<a href="http://appserver.example.com/foo/bar.html">foo` to `<a href="http://www.example.com/appserver/foo/bar.html"` making it accessible from outside.

`mod_proxy_html` was originally developed at WebPing, whose extensive [documentation](#) may be useful to users.



## ProxyHTMLBufSize Directive

<b>Description:</b>	Sets the buffer size increment for buffering inline scripts and stylesheets.
<b>Syntax:</b>	ProxyHTMLBufSize <i>bytes</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

In order to parse non-HTML content (stylesheets and scripts) embedded in HTML documents, mod\_proxy\_html has to read the entire script or stylesheet into a buffer. This buffer will be expanded as necessary to hold the largest script or stylesheet in a page, in increments of *bytes* as set by this directive.

The default is 8192, and will work well for almost all pages. However, if you know you're proxying pages containing stylesheets and/or scripts bigger than 8K (that is, for a single script or stylesheet, NOT in total), it will be more efficient to set a larger buffer size and avoid the need to resize the buffer dynamically during a request.



<b>Description:</b>	Specify a charset for mod_proxy_html output.
<b>Syntax:</b>	ProxyHTMLCharsetOut <i>Charset</i>   *
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

This selects an encoding for mod\_proxy\_html output. It should not normally be used, as any change from the default UTF-8 (Unicode - as used internally by libxml2) will impose an additional processing overhead. The special token ProxyHTMLCharsetOut \* will generate output using the same encoding as the input.

Note that this relies on [mod\\_xml2enc](#) being loaded.



## ProxyHTMLDocType Directive

<b>Description:</b>	Sets an HTML or XHTML document type declaration.
<b>Syntax:</b>	ProxyHTMLDocType <i>HTML XHTML</i> [ <i>Legacy</i> ] <b>OR</b> ProxyHTMLDocType <i>fpi</i> [ <i>SGML XML</i> ]
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

In the first form, documents will be declared as HTML 4.01 or XHTML 1.0 according to the option selected. This option also determines whether HTML or XHTML syntax is used for output. Note that the format of the documents coming from the backend server is immaterial: the parser will deal with it automatically. If the optional second argument is set to "Legacy", documents will be declared "Transitional", an option that may be necessary if you are proxying pre-1998 content or working with defective authoring/publishing tools.

In the second form, it will insert your own FPI. The optional second argument determines whether SGML/HTML or XML/XHTML syntax will be used.

The default is changed to omitting any FPI, on the grounds that no FPI is better than a bogus one. If your backend generates decent HTML or XHTML, set it accordingly.

If the first form is used, mod\_proxy\_html will also clean up the HTML to the specified standard. It cannot fix every error, but it will strip out bogus elements and attributes. It will also optionally log other errors at [LogLevel](#) Debug.



## ProxyHTMLModule Directive

<b>Description:</b>	Turns the proxy_html filter on or off.
<b>Syntax:</b>	ProxyHTMLEnable <i>On Off</i>
<b>Default:</b>	ProxyHTMLEnable Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party module for earlier 2.x versions.

A simple switch to enable or disable the proxy\_html filter. If `mod_xml2enc` is loaded it will also automatically set up internationalisation support.

Note that the proxy\_html filter will only act on HTML data (Content-Type text/html or application/xhtml+xml) and when the data are proxied. You can override this (at your own risk) by setting the `PROXY_HTML_FORCE` environment variable.



## ProxyHTMLEvents Directive

<b>Description:</b>	Specify attributes to treat as scripting events.
<b>Syntax:</b>	<code>ProxyHTMLEvents attribute [attribute ...]</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

Specifies one or more attributes to treat as scripting events and apply `ProxyHTMLURLMaps` to where enabled. You can specify any number of attributes in one or more `ProxyHTMLEvents` directives.

Normally you'll set this globally. If you set `ProxyHTMLEvents` in more than one scope so that one overrides the other, you'll need to specify a complete set in each of those scopes.

A default configuration is supplied in `proxy-html.conf` and defines the events in standard HTML 4 and XHTML 1.



## ProxyHTMLExtended Directive

<b>Description:</b>	Determines whether to fix links in inline scripts, stylesheets, and scripting events.
<b>Syntax:</b>	ProxyHTMLExtended <i>On Off</i>
<b>Default:</b>	ProxyHTMLExtended Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

Set to `Off`, HTML links are rewritten according to the `ProxyHTMLURLMap` directives, but links appearing in Javascript and CSS are ignored.

Set to `On`, all scripting events (as determined by `ProxyHTMLEvents`) and embedded scripts or stylesheets are also processed by the `ProxyHTMLURLMap` rules, according to the flags set for each rule. Since this requires more parsing, performance will be best if you only enable it when strictly necessary.

You'll also need to take care over patterns matched, since the parser has no knowledge of what is a URL within an embedded script or stylesheet. In particular, extended matching of `/` is likely to lead to false matches.



## ProxyHTMLFixups Directive

<b>Description:</b>	Fixes for simple HTML errors.
<b>Syntax:</b>	<code>ProxyHTMLFixups [lowercase] [dospath] [reset]</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

This directive takes one to three arguments as follows:

- lowercase Urls are rewritten to lowercase
- dospath Backslashes in URLs are rewritten to forward slashes.
- reset Unset any options set at a higher level in the configuration.

Take care when using these. The fixes will correct certain authoring mistakes, but risk also erroneously fixing links that were correct to start with. Only use them if you know you have a broken backend server.



## ProxyHTMLInterp Directive

<b>Description:</b>	Enables per-request interpolation of <code>ProxyHTMLURLMap</code> rules.
<b>Syntax:</b>	<code>ProxyHTMLInterp On Off</code>
<b>Default:</b>	<code>ProxyHTMLInterp Off</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_proxy_html</code>
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

This enables per-request interpolation in `ProxyHTMLURLMap` to- and from- patterns.

If interpolation is not enabled, all rules are pre-compiled at startup. With interpolation, they must be re-compiled for every request, which implies an extra processing overhead. It should therefore be enabled only when necessary.



## ProxyHTMLLinks Directive

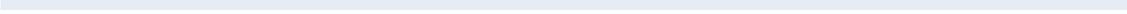
<b>Description:</b>	Specify HTML elements that have URL attributes to be rewritten.
<b>Syntax:</b>	<code>ProxyHTMLLinks <i>element attribute</i> [<i>attribute2 ...</i>]</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

Specifies elements that have URL attributes that should be rewritten using standard [ProxyHTMLURLMaps](#). You will need one ProxyHTMLLinks directive per element, but it can have any number of attributes.

Normally you'll set this globally. If you set ProxyHTMLLinks in more than one scope so that one overrides the other, you'll need to specify a complete set in each of those scopes.

A default configuration is supplied in `proxy-html.conf` and defines the HTML links for standard HTML 4 and XHTML 1.

```
Examples from proxy-html.conf  
  
ProxyHTMLLinks a href  
ProxyHTMLLinks area href  
ProxyHTMLLinks link href  
ProxyHTMLLinks img src longdesc usemap  
ProxyHTMLLinks object classid codebase data usemap  
ProxyHTMLLinks q cite  
ProxyHTMLLinks blockquote cite  
ProxyHTMLLinks ins cite  
ProxyHTMLLinks del cite  
ProxyHTMLLinks form action  
ProxyHTMLLinks input src usemap  
ProxyHTMLLinks head profile  
ProxyHTMLLinks base href  
ProxyHTMLLinks script src for
```



## ProxyHTMLMeta Directive

<b>Description:</b>	Turns on or off extra pre-parsing of metadata in HTML <head> sections.
<b>Syntax:</b>	ProxyHTMLMeta <i>On Off</i>
<b>Default:</b>	ProxyHTMLMeta Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party module for earlier 2.x versions.

This turns on or off pre-parsing of metadata in HTML <head> sections.

If not required, turning ProxyHTMLMeta Off will give a small performance boost by skipping this parse step. However, it is sometimes necessary for internationalisation to work correctly.

ProxyHTMLMeta has two effects. Firstly and most importantly it enables detection of character encodings declared in the form

```
<meta http-equiv="Content-Type" content="text/html
```

or, in the case of an XHTML document, an XML declaration. It is NOT required if the charset is declared in a real HTTP header (which is always preferable) from the backend server, nor if the document is *utf-8* (unicode) or a subset such as ASCII. You may also be able to dispense with it where documents use a default declared using [xml2EncDefault](#), but that risks propagating an incorrect declaration. A [ProxyHTMLCharsetOut](#) can remove that risk, but is likely to be a bigger processing overhead than enabling ProxyHTMLMeta.

The other effect of enabling ProxyHTMLMeta is to parse all <meta

`http-equiv=...` declarations and convert them to real HTTP headers, in keeping with the original purpose of this form of the HTML `<meta>` element.

### **Warning**

Because ProxyHTMLMeta promotes **all** `http-equiv` elements to HTTP headers, it is important that you only enable it in cases where you trust the HTML content as much as you trust the upstream server. If the HTML is controlled by bad actors, it will be possible for them to inject arbitrary, possibly malicious, HTTP headers into your server's responses.



## ProxyHTMLStripComments Directive

<b>Description:</b>	Determines whether to strip HTML comments.
<b>Syntax:</b>	ProxyHTMLStripComments <i>On Off</i>
<b>Default:</b>	ProxyHTMLStripComments Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party for earlier 2.x versions

This directive will cause mod\_proxy\_html to strip HTML comments. Note that this will also kill off any scripts or styles embedded in comments (a bogosity introduced in 1995/6 with Netscape 2 for the benefit of then-older browsers, but still in use today). It may also interfere with comment-based processors such as SSI or ESI: be sure to run any of those *before* mod\_proxy\_html in the filter chain if stripping comments!



## ProxyHTMLURLMap Directive

<b>Description:</b>	Defines a rule to rewrite HTML links
<b>Syntax:</b>	ProxyHTMLURLMap <i>from-pattern to-pattern [flags] [cond]</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Base
<b>Module:</b>	mod_proxy_html
<b>Compatibility:</b>	Version 2.4 and later; available as a third-party module for earlier 2.x versions.

This is the key directive for rewriting HTML links. When parsing a document, whenever a link target matches *from-pattern*, the matching portion will be rewritten to *to-pattern*, as modified by any flags supplied and by the [ProxyHTMLExtended](#) directive. Only the elements specified using the [ProxyHTMLLinks](#) directive will be considered as HTML links.

The optional third argument may define any of the following **Flags**. Flags are case-sensitive.

- h**  
Ignore HTML links (pass through unchanged)
- e**  
Ignore scripting events (pass through unchanged)
- c**  
Pass embedded script and style sections through untouched.
- L**  
Last-match. If this rule matches, no more rules are applied (note that this happens automatically for HTML links).
- I**  
Opposite to L. Overrides the one-change-only default

behaviour with HTML links.

## **R**

Use Regular Expression matching-and-replace. `from-pattern` is a regexp, and `to-pattern` a replacement string that may be based on the regexp. Regexp memory is supported: you can use brackets `()` in the `from-pattern` and retrieve the matches with `$1` to `$9` in the `to-pattern`.

If `R` is not set, it will use string-literal search-and-replace. The logic is *starts-with* in HTML links, but *contains* in scripting events and embedded script and style sections.

## **x**

Use POSIX extended Regular Expressions. Only applicable with `R`.

## **i**

Case-insensitive matching. Only applicable with `R`.

## **n**

Disable regexp memory (for speed). Only applicable with `R`.

## **s**

Line-based regexp matching. Only applicable with `R`.

## **^**

Match at start only. This applies only to string matching (not regexps) and is irrelevant to HTML links.

## **\$**

Match at end only. This applies only to string matching (not regexps) and is irrelevant to HTML links.

## **V**

Interpolate environment variables in `to-pattern`. A string of

the form `${varname | default}` will be replaced by the value of environment variable `varname`. If that is unset, it is replaced by `default`. The `| default` is optional.

NOTE: interpolation will only be enabled if `ProxyHTMLInterp` is *On*.

## V

Interpolate environment variables in `from-pattern`. Patterns supported are as above.

NOTE: interpolation will only be enabled if `ProxyHTMLInterp` is *On*.

The optional fourth **cond** argument defines a condition that will be evaluated per Request, provided `ProxyHTMLInterp` is *On*. If the condition evaluates FALSE the map will not be applied in this request. If TRUE, or if no condition is defined, the map is applied.

A **cond** is evaluated by the [Expression Parser](#). In addition, the simpler syntax of conditions in `mod_proxy_html` 3.x for HTTPD 2.0 and 2.2 is also supported.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_http`

<b>Description:</b>	HTTP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_http_module</code>
<b>Source File:</b>	<code>mod_proxy_http.c</code>

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides the features used for proxying HTTP and HTTPS requests. [mod\\_proxy\\_http](#) supports HTTP/0.9, HTTP/1.0 and HTTP/1.1. It does *not* provide any caching abilities. If you want to set up a caching proxy, you might want to use the additional service of the [mod\\_cache](#) module.

Thus, in order to get the ability of handling HTTP proxy requests, [mod\\_proxy](#) and [mod\\_proxy\\_http](#) have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

[mod\\_proxy\\_connect](#)



---

In addition to the configuration directives that control the behaviour of `mod_proxy`, there are a number of *environment variables* that control the HTTP protocol provider. Environment variables below that don't specify specific values are enabled when set to any value.

### **proxy-sendextracrlf**

Causes proxy to send an extra CR-LF newline on the end of a request. This is a workaround for a bug in some browsers.

### **force-proxy-request-1.0**

Forces the proxy to send requests to the backend as HTTP/1.0 and disables HTTP/1.1 features.

### **proxy-nokeepalive**

Forces the proxy to close the backend connection after each request.

### **proxy-chain-auth**

If the proxy requires authentication, it will read and consume the proxy authentication credentials sent by the client. With *proxy-chain-auth* it will *also* forward the credentials to the next proxy in the chain. This may be necessary if you have a chain of proxies that share authentication information. **Security Warning:** Do not set this unless you know you need it, as it forwards sensitive information!

### **proxy-sendcl**

HTTP/1.0 required all HTTP requests that include a body (e.g. POST requests) to include a *Content-Length* header. This environment variable forces the Apache proxy to send this header to the backend server, regardless of what the Client sent to the proxy. It ensures compatibility when proxying for an HTTP/1.0 or unknown backend. However, it may require the entire request to be buffered by the proxy, so it becomes very inefficient for large requests.

### **proxy-sendchunks or proxy-sendchunked**

This is the opposite of *proxy-sendcl*. It allows request bodies to be sent to the backend using chunked transfer encoding. This allows the request to be efficiently streamed, but requires that the backend server supports HTTP/1.1.

### **proxy-interim-response**

This variable takes values RFC (the default) or Suppress. Earlier httpd versions would suppress HTTP interim (1xx) responses sent from the backend. This is technically a violation of the HTTP protocol. In practice, if a backend sends an interim response, it may itself be extending the protocol in a manner we know nothing about, or just broken. So this is now configurable: set `proxy-interim-response RFC` to be fully protocol compliant, or `proxy-interim-response Suppress` to suppress interim responses.

### **proxy-initial-not-pooled**

If this variable is set, no pooled connection will be reused if the client request is the initial request on the frontend connection. This avoids the "proxy: error reading status line from remote server" error message caused by the race condition that the backend server closed the pooled connection after the connection check by the proxy and before data sent by the proxy reached the backend. It has to be kept in mind that setting this variable downgrades performance, especially with HTTP/1.0 clients.



**Request Notes**

`mod_proxy_http` creates the following request notes for logging using the `%{VARNAME}`n format in [LogFormat](#) or [ErrorLogFormat](#):

**proxy-source-port**

The local port used for the connection to the backend server.

**proxy-status**

The HTTP status received from the backend server.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_http2`

<b>Description:</b>	HTTP/2 support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_http2_module</code>
<b>Source File:</b>	<code>mod_proxy_http2.c</code>

## Summary

[mod\\_proxy\\_http2](#) supports HTTP/2 only, it does *not* provide any downgrades to HTTP/1.1. This means that the backend needs to support HTTP/2 because HTTP/1.1 will not be used instead.

This module *requires* the service of [mod\\_proxy](#), so in order to get the ability of handling HTTP/2 proxy requests, [mod\\_proxy](#) and [mod\\_proxy\\_http2](#) need to be both loaded by the server.

[mod\\_proxy\\_http2](#) works with incoming fronted requests using HTTP/1.1 or HTTP/2. In both cases, requests proxied to the same backend are sent over a single TCP connection whenever possible (namely when the connection can be re-used).

Caveat: there will be no attempt to consolidate multiple HTTP/1.1 frontend requests (configured to be proxied to the same backend) into HTTP/2 streams belonging to the same HTTP/2 request. Each HTTP/1.1 frontend request will be proxied to the backend using a separate HTTP/2 request (trying to re-use the same TCP connection if possible).

This module relies on [libnghttp2](#) to provide the core http/2 engine.

## Warning

This module is experimental. Its behaviors, directives, and defaults are subject to more change from release to release relative to other

standard modules. Users are encouraged to consult the "CHANGES" file for potential updates.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_http2](#)

[mod\\_proxy](#)

[mod\\_proxy\\_connect](#)



## Basic Examples

The examples below demonstrate how to configure HTTP/2 for backend connections for a reverse proxy.

### HTTP/2 (TLS)

```
ProxyPass "/app" "h2://app.example.com"  
ProxyPassReverse "/app" "https://app.example.com"
```

### HTTP/2 (cleartext)

```
ProxyPass "/app" "h2c://app.example.com"  
ProxyPassReverse "/app" "http://app.example.com"
```

The schemes to configure above in `ProxyPassReverse` for reverse proxying h2 (or h2c) protocols are the usual `https` (resp. `http`) as expected/used by the user agent.



**Request Notes**

`mod_proxy_http` creates the following request notes for logging using the `%{VARNAME}`n format in [LogFormat](#) or [ErrorLogFormat](#):

**proxy-source-port**

The local port used for the connection to the backend server.

**proxy-status**

The HTTP/2 status received from the backend server.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_scgi`

<b>Description:</b>	SCGI gateway module for <code>mod_proxy</code>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_scgi_module</code>
<b>Source File:</b>	<code>mod_proxy_scgi.c</code>
<b>Compatibility:</b>	Available in version 2.2.14 and later

## Summary

This module *requires* the service of `mod_proxy`. It provides support for the [SCGI protocol, version 1](#).

Thus, in order to get the ability of handling the SCGI protocol, `mod_proxy` and `mod_proxy_scgi` have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

mod\_proxy

mod\_proxy balancer



Remember, in order to make the following examples work, you have to enable [mod\\_proxy](#) and [mod\\_proxy\\_scgi](#).

### Simple gateway

```
ProxyPass /scgi-bin/ scgi://localhost:4000/
```

The balanced gateway needs [mod\\_proxy\\_balancer](#) and at least one load balancer algorithm module, such as [mod\\_lbmethod\\_byrequests](#), in addition to the proxy modules listed above. [mod\\_lbmethod\\_byrequests](#) is the default, and will be used for this example configuration.

### Balanced gateway

```
ProxyPass "/scgi-bin/" "balancer://somecluster/"  
<Proxy "balancer://somecluster">  
    BalancerMember "scgi://localhost:4000"  
    BalancerMember "scgi://localhost:4001"  
</Proxy>
```



In addition to the configuration directives that control the behaviour of [mod\\_proxy](#), an *environment variable* may also control the SCGI protocol provider:

### **proxy-scgi-pathinfo**

By default [mod\\_proxy\\_scgi](#) will neither create nor export the `PATH_INFO` environment variable. This allows the backend SCGI server to correctly determine `SCRIPT_NAME` and `Script-URI` and be compliant with RFC 3875 section 3.3. If instead you need [mod\\_proxy\\_scgi](#) to generate a "best guess" for `PATH_INFO`, set this env-var. The variable must be set before [SetEnv](#) is effective. [SetEnvIf](#) can be used instead: `SetEnvIf Request_URI . proxy-scgi-pathinfo`



<b>Description:</b>	Enable or disable internal redirect responses from the backend
<b>Syntax:</b>	ProxySCGIInternalRedirect On Off  <i>Headername</i>
<b>Default:</b>	ProxySCGIInternalRedirect On
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_scgi
<b>Compatibility:</b>	The <i>Headername</i> feature is available in version 2.4.13 and later

The `ProxySCGIInternalRedirect` enables the backend to internally redirect the gateway to a different URL. This feature originates in `mod_cgi`, which internally redirects the response if the response status is OK (200) and the response contains a `Location` (or configured alternate header) and its value starts with a slash (/). This value is interpreted as a new local URL that Apache httpd internally redirects to.

`mod_proxy_scgi` does the same as `mod_cgi` in this regard, except that you can turn off the feature or specify the use of a header other than `Location`.

### Example

```
ProxySCGIInternalRedirect Off

# Django and some other frameworks will fully qualify "local URI
# set by the application, so an alternate header must be used.
<Location /django-app/>
    ProxySCGIInternalRedirect X-Location
</Location>
```



<b>Description:</b>	Enable evaluation of <i>X-Sendfile</i> pseudo response header
<b>Syntax:</b>	<code>ProxySCGISendfile On Off Headername</code>
<b>Default:</b>	<code>ProxySCGISendfile Off</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_proxy_scgi</code>

The `ProxySCGISendfile` directive enables the SCGI backend to let files be served directly by the gateway. This is useful for performance purposes — httpd can use `sendfile` or other optimizations, which are not possible if the file comes over the backend socket. Additionally, the file contents are not transmitted twice.

The `ProxySCGISendfile` argument determines the gateway behaviour:

### **off**

No special handling takes place.

### **On**

The gateway looks for a backend response header called `X-Sendfile` and interprets the value as the filename to serve. The header is removed from the final response headers. This is equivalent to `ProxySCGISendfile X-Sendfile`.

### **anything else**

Similar to `On`, but instead of the hardcoded header name `X-Sendfile`, the argument is used as the header name.

### **Example**

```
# Use the default header (X-Sendfile)
ProxySCGISendfile On
```

```
# Use a different header
ProxySCGISendfile X-Send-Static
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_proxy_wstunnel`

<b>Description:</b>	Websockets support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_wstunnel_module</code>
<b>Source File:</b>	<code>mod_proxy_wstunnel.c</code>
<b>Compatibility:</b>	Available in httpd 2.4.5 and later

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides support for the tunnelling of web socket connections to a backend websockets server. The connection is automatically upgraded to a websocket connection:

### HTTP Response

```
Upgrade: WebSocket  
Connection: Upgrade
```

Proxying requests to a websockets server like `echo.websocket.org` can be done using the [ProxyPass](#) directive:

```
ProxyPass "/ws2/" "ws://echo.websocket.org/"  
ProxyPass "/wss2/" "wss://echo.websocket.org/"
```

Load balancing for multiple backends can be achieved using [mod\\_proxy\\_balancer](#).

In fact the module can be used to upgrade to other protocols, you can set the upgrade parameter in the [ProxyPass](#) directive to allow the module to accept other protocol. NONE means you bypass the check for the header but still upgrade to WebSocket. ANY means that

Upgrade will read in the request headers and use in the response  
Upgrade



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_proxy](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_ratelimit

<b>Description:</b>	Bandwidth Rate Limiting for Clients
<b>Status:</b>	Extension
<b>Module Identifier:</b>	ratelimit_module
<b>Source File:</b>	mod_ratelimit.c
<b>Compatibility:</b>	rate-initial-burst available in httpd 2.4.24 and later.

## Summary

Provides a filter named `RATE_LIMIT` to limit client bandwidth. The throttling is applied to each HTTP response while it is transferred to the client, and not aggregated at IP/client level. The connection speed to be simulated is specified, in KiB/s, using the environment variable `rate-limit`.

Optionally, an initial amount of burst data, in KiB, may be configured to be passed at full speed before throttling to the specified rate limit. This value is optional, and is set using the environment variable `rate-initial-burst`.

## Example Configuration

```
<Location "/downloads">
  SetOutputFilter RATE_LIMIT
  SetEnv rate-limit 400
  SetEnv rate-initial-burst 512
</Location>
```

If the value specified for `rate-limit` causes integer overflow, the rate-limited will be disabled. If the value specified for `rate-limit-burst` causes integer overflow, the burst will be disabled.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_reflector

<b>Description:</b>	Reflect a request body as a response via the output filter stack.
<b>Status:</b>	Base
<b>Module Identifier:</b>	reflector_module
<b>Source File:</b>	mod_reflector.c
<b>Compatibility:</b>	Version 2.3 and later

## Summary

This module allows request bodies to be reflected back to the client, in the process passing the request through the output filter stack. A suitably configured chain of filters can be used to transform the request into a response. This module can be used to turn an output filter into an HTTP service.



## Examples

### Compression service

Pass the request body through the DEFLATE filter to compress the body. This request requires a Content-Encoding request header containing "gzip" for the filter to return compressed data.

```
<Location "/compress">  
  SetHandler reflector  
  SetOutputFilter DEFLATE  
</Location>
```

### Image downsampling service

Pass the request body through an image downsampling filter, and reflect the results to the caller.

```
<Location "/downsample">  
  SetHandler reflector  
  SetOutputFilter DOWNSAMPLE  
</Location>
```



<b>Description:</b>	Reflect an input header to the output headers
<b>Syntax:</b>	ReflectorHeader <i>inputheader</i> <i>[outputheader]</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_reflector

This directive controls the reflection of request headers to the response. The first argument is the name of the request header to copy. If the optional second argument is specified, it will be used as the name of the response header, otherwise the original request header name will be used.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_remoteip

<b>Description:</b>	Replaces the original client IP address for the connection with the useragent IP address list presented by a proxies or a load balancer via the request headers.
<b>Status:</b>	Base
<b>Module Identifier:</b>	remoteip_module
<b>Source File:</b>	mod_remoteip.c

## Summary

This module is used to treat the useragent which initiated the request as the originating useragent as identified by httpd for the purposes of authorization and logging, even where that useragent is behind a load balancer, front end server, or proxy server.

The module overrides the client IP address for the connection with the useragent IP address reported in the request header configured with the [RemoteIPHeader](#) directive.

Once replaced as instructed, this overridden useragent IP address is then used for the [mod\\_authz\\_host Require ip](#) feature, is reported by [mod\\_status](#), and is recorded by [mod\\_log\\_config](#) %a and [core](#) %a format strings. The underlying client IP of the connection is available in the %[c](#)a format string.

It is critical to only enable this behavior from intermediate hosts (proxies, etc) which are trusted by this server, since it is trivial for the remote useragent to impersonate another useragent.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_authz\\_host](#)

[mod\\_status](#)

[mod\\_log\\_config](#)



## Remote IP Processing

Apache by default identifies the useragent with the connection's `client_ip` value, and the connection `remote_host` and `remote_logname` are derived from this value. These fields play a role in authentication, authorization and logging and other purposes by other loadable modules.

`mod_remoteip` overrides the client IP of the connection with the advertised useragent IP as provided by a proxy or load balancer, for the duration of the request. A load balancer might establish a long lived keepalive connection with the server, and each request will have the correct useragent IP, even though the underlying client IP address of the load balancer remains unchanged.

When multiple, comma delimited useragent IP addresses are listed in the header value, they are processed in Right-to-Left order. Processing halts when a given useragent IP address is not trusted to present the preceding IP address. The header field is updated to this remaining list of unconfirmed IP addresses, or if all IP addresses were trusted, this header is removed from the request altogether.

In overriding the client IP, the module stores the list of intermediate hosts in a `remoteip-proxy-ip-list` note, which [mod\\_log\\_config](#) can record using the `%{remoteip-proxy-ip-list}` format token. If the administrator needs to store this as an additional header, this same value can also be recording as a header using the directive [RemoteIPProxiesHeader](#).

### IPv4-over-IPv6 Mapped Addresses

As with `httpd` in general, any IPv4-over-IPv6 mapped addresses are recorded in their IPv4 representation.

### Internal (Private) Addresses

All internal addresses 10/8, 172.16/12, 192.168/16, 169.254/16 and 127/8 blocks (and IPv6 addresses outside of the public 2000::/3 block) are only evaluated by mod\_remoteip when [RemoteIPInternalProxy](#) internal (intranet) proxies are registered.



<b>Description:</b>	Declare the header field which should be parsed for useragent IP addresses
<b>Syntax:</b>	RemoteIPHeader <i>header-field</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_remoteip

The [RemoteIPHeader](#) directive triggers [mod\\_remoteip](#) to treat the value of the specified *header-field* header as the useragent IP address, or list of intermediate useragent IP addresses, subject to further configuration of the [RemoteIPInternalProxy](#) and [RemoteIPTrustedProxy](#) directives. Unless these other directives are used, [mod\\_remoteip](#) will trust all hosts presenting a [RemoteIPHeader](#) IP value.

### Internal (Load Balancer) Example

```
RemoteIPHeader X-Client-IP
```

### Proxy Example

```
RemoteIPHeader X-Forwarded-For
```



## RemoteIPInternalProxy Directive

<b>Description:</b>	Declare client intranet IP addresses trusted to present the RemoteIPHeader value
<b>Syntax:</b>	<code>RemoteIPInternalProxy proxy-ip proxy-ip/subnet hostname ...</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_remoteip

The `RemoteIPInternalProxy` directive adds one or more addresses (or address blocks) to trust as presenting a valid RemoteIPHeader value of the useragent IP. Unlike the `RemoteIPTrustedProxy` directive, any IP address presented in this header, including private intranet addresses, are trusted when passed from these proxies.

### Internal (Load Balancer) Example

```
RemoteIPHeader X-Client-IP
RemoteIPInternalProxy 10.0.2.0/24
RemoteIPInternalProxy gateway.localdomain
```



<b>Description:</b>	Declare client intranet IP addresses trusted to present the RemoteIPHeader value
<b>Syntax:</b>	RemoteIPInternalProxyList <i>filename</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_remoteip

The [RemoteIPInternalProxyList](#) directive specifies a file parsed at startup, and builds a list of addresses (or address blocks) to trust as presenting a valid RemoteIPHeader value of the useragent IP.

The '#' hash character designates a comment line, otherwise each whitespace or newline separated entry is processed identically to the [RemoteIPInternalProxy](#) directive.

### Internal (Load Balancer) Example

```
RemoteIPHeader X-Client-IP
RemoteIPInternalProxyList conf/trusted-proxies.lst
```

### conf/trusted-proxies.lst contents

```
# Our internally trusted proxies;
10.0.2.0/24           #Everyone in the testing group
gateway.localdomain #The front end balancer
```



<b>Description:</b>	Declare the header field which will record all intermediate IP addresses
<b>Syntax:</b>	<code>RemoteIPProxiesHeader <i>HeaderFieldName</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_remoteip</code>

The [RemoteIPProxiesHeader](#) directive specifies a header into which `mod_remoteip` will collect a list of all of the intermediate client IP addresses trusted to resolve the useragent IP of the request. Note that intermediate [RemoteIPTrustedProxy](#) addresses are recorded in this header, while any intermediate [RemoteIPInternalProxy](#) addresses are discarded.

### Example

```
RemoteIPHeader X-Forwarded-For
RemoteIPProxiesHeader X-Forwarded-By
```



## RemoteIPTrustedProxy Directive

<b>Description:</b>	Declare client intranet IP addresses trusted to present the RemoteIPHeader value
<b>Syntax:</b>	RemoteIPTrustedProxy <i>proxy-ip proxy-ip/subnet hostname ...</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_remoteip

The [RemoteIPTrustedProxy](#) directive adds one or more addresses (or address blocks) to trust as presenting a valid RemoteIPHeader value of the useragent IP. Unlike the [RemoteIPInternalProxy](#) directive, any intranet or private IP address reported by such proxies, including the 10/8, 172.16/12, 192.168/16, 169.254/16 and 127/8 blocks (or outside of the IPv6 public 2000::/3 block) are not trusted as the useragent IP, and are left in the [RemoteIPHeader](#) header's value.

```
Trusted (Load Balancer) Example  
RemoteIPHeader X-Forwarded-For  
RemoteIPTrustedProxy 10.0.2.16/28  
RemoteIPTrustedProxy proxy.example.com
```



<b>Description:</b>	Declare client intranet IP addresses trusted to present the RemoteIPHeader value
<b>Syntax:</b>	RemoteIPTrustedProxyList <i>filename</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_remoteip

The [RemoteIPTrustedProxyList](#) directive specifies a file parsed at startup, and builds a list of addresses (or address blocks) to trust as presenting a valid RemoteIPHeader value of the useragent IP.

The '#' hash character designates a comment line, otherwise each whitespace or newline separated entry is processed identically to the [RemoteIPTrustedProxy](#) directive.

### Trusted (Load Balancer) Example

```
RemoteIPHeader X-Forwarded-For
RemoteIPTrustedProxyList conf/trusted-proxies.lst
```

### conf/trusted-proxies.lst contents

```
# Identified external proxies;
192.0.2.16/28 #wap phone group of proxies
proxy.isp.example.com #some well known ISP
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_reqtimeout

<b>Description:</b>	Set timeout and minimum data rate for receiving requests
<b>Status:</b>	Extension
<b>Module Identifier:</b>	reqtimeout_module
<b>Source File:</b>	mod_reqtimeout.c
<b>Compatibility:</b>	Available in Apache HTTPD 2.2.15 and later



## Examples

1. Allow 10 seconds to receive the request including the headers and 30 seconds for receiving the request body:

```
RequestReadTimeout header=10 body=30
```

2. Allow at least 10 seconds to receive the request body. If the client sends data, increase the timeout by 1 second for every 1000 bytes received, with no upper limit for the timeout (except for the limit given indirectly by [LimitRequestBody](#)):

```
RequestReadTimeout body=10,MinRate=1000
```

3. Allow at least 10 seconds to receive the request including the headers. If the client sends data, increase the timeout by 1 second for every 500 bytes received. But do not allow more than 30 seconds for the request including the headers:

```
RequestReadTimeout header=10-30,MinRate=
```

4. Usually, a server should have both header and body timeouts configured. If a common configuration is used for http and https virtual hosts, the timeouts should not be set too low:

```
RequestReadTimeout header=20-40,MinRate=
```



## RequestReadTimeout Directive

<b>Description:</b>	Set timeout values for receiving request headers and body from client.
<b>Syntax:</b>	<code>RequestReadTimeout [header=<i>timeout</i> [-<i>maxtimeout</i>]] [,MinRate=<i>rate</i>] [body=<i>timeout</i> [-<i>maxtimeout</i>]] [,MinRate=<i>rate</i>]</code>
<b>Default:</b>	header=20-40,MinRate=500 body=20,MinRate=500
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_reqtimeout
<b>Compatibility:</b>	Available in version 2.2.15 and later; defaulted to disabled in version 2.3.14 and earlier.

This directive can set various timeouts for receiving the request headers and the request body from the client. If the client fails to send headers or body within the configured time, a 408 `REQUEST TIME OUT` error is sent.

For SSL virtual hosts, the header timeout values include the time needed to do the initial SSL handshake. If the user's browser is configured to query certificate revocation lists and the CRL server is not reachable, the initial SSL handshake may take a significant time until the browser gives up waiting for the CRL. Therefore the header timeout values should not be set to very low values for SSL virtual hosts. The body timeout values include the time needed for SSL renegotiation (if necessary).

When an [AcceptFilter](#) is in use (usually the case on Linux and FreeBSD), the socket is not sent to the server process before at least one byte (or the whole request for `httpready`) is received. The header timeout configured with `RequestReadTimeout` is

only effective after the server process has received the socket.

For each of the two timeout types (header or body), there are three ways to specify the timeout:

- **Fixed timeout value:**

```
type=timeout
```

The time in seconds allowed for reading all of the request headers or body, respectively. A value of 0 means no limit.

- **Disable module for a vhost:**

```
header=0 body=0
```

This disables [mod\\_reqtimeout](#) completely.

- **Timeout value that is increased when data is received:**

```
type=timeout,MinRate=data_rate
```

Same as above, but whenever data is received, the timeout value is increased according to the specified minimum data rate (in bytes per second).

- **Timeout value that is increased when data is received, with an upper bound:**

```
type=timeout-maxtimeout,MinRate=data_rate
```

Same as above, but the timeout will not be increased above the second value of the specified timeout range.

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_request

<b>Description:</b>	Filters to handle and make available HTTP request bodies
<b>Status:</b>	Base
<b>Module Identifier:</b>	request_module
<b>Source File:</b>	mod_request.c
<b>Compatibility:</b>	Available in Apache 2.3 and later



<b>Description:</b>	Keep the request body instead of discarding it up to the specified maximum size, for potential use by filters such as <code>mod_include</code> .
<b>Syntax:</b>	<code>KeptBodySize</code> <i>maximum size in bytes</i>
<b>Default:</b>	<code>KeptBodySize 0</code>
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_request</code>

Under normal circumstances, request handlers such as the default handler for static files will discard the request body when it is not needed by the request handler. As a result, filters such as `mod_include` are limited to making GET requests only when including other URLs as subrequests, even if the original request was a POST request, as the discarded request body is no longer available once filter processing is taking place.

When this directive has a value greater than zero, request handlers that would otherwise discard request bodies will instead set the request body aside for use by filters up to the maximum size specified. In the case of the `mod_include` filter, an attempt to POST a request to the static shtml file will cause any subrequests to be POST requests, instead of GET requests as before.

This feature makes it possible to break up complex web pages and web applications into small individual components, and combine the components and the surrounding web page structure together using `mod_include`. The components can take the form of CGI programs, scripted languages, or URLs reverse proxied into the URL space from another server using `mod_proxy`.

**Note:** Each request set aside has to be set aside in temporary RAM until the request is complete. As a result, care should be

taken to ensure sufficient RAM is available on the server to support the intended load. Use of this directive should be limited to where needed on targeted parts of your URL space, and with the lowest possible value that is still big enough to hold a request body.

If the request size sent by the client exceeds the maximum size allocated by this directive, the server will return 413 Request Entity Too Large.

## See also

- [mod\\_include](#) documentation
- [mod\\_auth\\_form](#) documentation

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_rewrite`

<b>Description:</b>	Provides a rule-based rewriting engine to rewrite requested URLs on the fly
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>rewrite_module</code>
<b>Source File:</b>	<code>mod_rewrite.c</code>

## Summary

The `mod_rewrite` module uses a rule-based rewriting engine, based on a PCRE regular-expression parser, to rewrite requested URLs on the fly. By default, `mod_rewrite` maps a URL to a filesystem path. However, it can also be used to redirect one URL to another URL, or to invoke an internal proxy fetch.

`mod_rewrite` provides a flexible and powerful way to manipulate URLs using an unlimited number of rules. Each rule can have an unlimited number of attached rule conditions, to allow you to rewrite URL based on server variables, environment variables, HTTP headers, or time stamps.

`mod_rewrite` operates on the full URL path, including the path-info section. A rewrite rule can be invoked in `httpd.conf` or in `.htaccess`. The path generated by a rewrite rule can include a query string, or can lead to internal sub-processing, external request redirection, or internal proxy throughput.

Further details, discussion, and examples, are provided in the [detailed `mod\_rewrite` documentation](#).



## Logging

`mod_rewrite` offers detailed logging of its actions at the `trace1` to `trace8` log levels. The log level can be set specifically for `mod_rewrite` using the `LogLevel` directive: Up to level `debug`, no actions are logged, while `trace8` means that practically all actions are logged.

Using a high trace log level for `mod_rewrite` will slow down your Apache HTTP Server dramatically! Use a log level higher than `trace2` only for debugging!

### Example

```
LogLevel alert rewrite:trace3
```

### RewriteLog

Those familiar with earlier versions of `mod_rewrite` will no doubt be looking for the `RewriteLog` and `RewriteLogLevel` directives. This functionality has been completely replaced by the new per-module logging configuration mentioned above.

To get just the `mod_rewrite`-specific log messages, pipe the log file through `grep`:

```
tail -f error_log|fgrep '[rewrite:]'
```



<b>Description:</b>	Sets the base URL for per-directory rewrites
<b>Syntax:</b>	RewriteBase <i>URL-path</i>
<b>Default:</b>	None
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteBase` directive specifies the URL prefix to be used for per-directory (htaccess) `RewriteRule` directives that substitute a relative path.

This directive is *required* when you use a relative path in a substitution in per-directory (htaccess) context unless any of the following conditions are true:

- The original request, and the substitution, are underneath the `DocumentRoot` (as opposed to reachable by other means, such as `Alias`).
- The *filesystem* path to the directory containing the `RewriteRule`, suffixed by the relative substitution is also valid as a URL path on the server (this is rare).
- In Apache HTTP Server 2.4.16 and later, this directive may be omitted when the request is mapped via `Alias` or `mod_userdir`.

In the example below, `RewriteBase` is necessary to avoid rewriting to `http://example.com/opt/myapp-1.2.3/welcome.html` since the resource was not relative to the document root. This misconfiguration would normally cause the server to look for an "opt" directory under the document root.

```
DocumentRoot "/var/www/example.com"  
AliasMatch "^/myapp" "/opt/myapp-1.2.3"  
<Directory "/opt/myapp-1.2.3">  
    RewriteEngine On  
    RewriteBase "/myapp/"  
    RewriteRule "^index\.html$" "welcome.ht  
</Directory>
```



<b>Description:</b>	Defines a condition under which rewriting will take place
<b>Syntax:</b>	<code>RewriteCond <i>TestString</i> <i>CondPattern</i> [<i>flags</i>]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteCond` directive defines a rule condition. One or more `RewriteCond` can precede a `RewriteRule` directive. The following rule is then only used if both the current state of the URI matches its pattern, **and** if these conditions are met.

`TestString` is a string which can contain the following expanded constructs in addition to plain text:

- **RewriteRule backreferences:** These are backreferences of the form `$N` ( $0 \leq N \leq 9$ ). `$1` to `$9` provide access to the grouped parts (in parentheses) of the pattern, from the `RewriteRule` which is subject to the current set of `RewriteCond` conditions. `$0` provides access to the whole string matched by that pattern.
- **RewriteCond backreferences:** These are backreferences of the form `%N` ( $0 \leq N \leq 9$ ). `%1` to `%9` provide access to the grouped parts (again, in parentheses) of the pattern, from the last matched `RewriteCond` in the current set of conditions. `%0` provides access to the whole string matched by that pattern.
- **RewriteMap expansions:** These are expansions of the form `${mapname:key|default}`. See [the documentation for RewriteMap](#) for more details.

- **Server-Variables:** These are variables of the form `%{NAME_OF_VARIABLE}` where *NAME\_OF\_VARIABLE* can be a string taken from the following list:

<b>HTTP headers:</b>	<b>connection &amp; request:</b>
HTTP_ACCEPT	AUTH_TYPE
HTTP_COOKIE	CONN_REMOTE_ADDR
HTTP_FORWARDED	CONTEXT_PREFIX
HTTP_HOST	CONTEXT_DOCUMENT_ROOT
HTTP_PROXY_CONNECTION	IPV6
HTTP_REFERER	PATH_INFO
HTTP_USER_AGENT	QUERY_STRING
	REMOTE_ADDR
	REMOTE_HOST
	REMOTE_IDENT
	REMOTE_PORT
	REMOTE_USER
	REQUEST_METHOD
	SCRIPT_FILENAME
<b>server internals:</b>	<b>date and time:</b>
DOCUMENT_ROOT	TIME_YEAR
SCRIPT_GROUP	TIME_MON
SCRIPT_USER	TIME_DAY
SERVER_ADDR	TIME_HOUR
SERVER_ADMIN	TIME_MIN
SERVER_NAME	TIME_SEC
SERVER_PORT	TIME_WDAY
SERVER_PROTOCOL	TIME
SERVER_SOFTWARE	

These variables all correspond to the similarly named HTTP MIME-headers, C variables of the Apache HTTP Server or struct tm fields of the Unix system. Most are documented [here](#) or elsewhere in the Manual or in the CGI specification.

SERVER\_NAME and SERVER\_PORT depend on the values of [UseCanonicalName](#) and [UseCanonicalPhysicalPort](#) respectively.

Those that are special to mod\_rewrite include those below.

### **API\_VERSION**

This is the version of the Apache httpd module API (the internal interface between server and module) in the current httpd build, as defined in include/ap\_mmn.h. The module API version corresponds to the version of Apache httpd in use (in the release version of Apache httpd 1.3.14, for instance, it is 19990320:10), but is mainly of interest to module authors.

### **CONN\_REMOTE\_ADDR**

Since 2.4.8: The peer IP address of the connection (see the [mod\\_remoteip](#) module).

### **HTTPS**

Will contain the text "on" if the connection is using SSL/TLS, or "off" otherwise. (This variable can be safely used regardless of whether or not [mod\\_ssl](#) is loaded).

### **IS\_SUBREQ**

Will contain the text "true" if the request currently being processed is a sub-request, "false" otherwise. Sub-requests may be generated by modules that need to resolve additional files or URIs in order to complete their tasks.

### **REMOTE\_ADDR**

The IP address of the remote host (see the [mod\\_remoteip](#) module).

### **REQUEST\_FILENAME**

The full local filesystem path to the file or script matching

the request, if this has already been determined by the server at the time `REQUEST_FILENAME` is referenced. Otherwise, such as when used in virtual host context, the same value as `REQUEST_URI`. Depending on the value of [AcceptPathInfo](#), the server may have only used some leading components of the `REQUEST_URI` to map the request to a file.

### **REQUEST\_SCHEME**

Will contain the scheme of the request (usually "http" or "https"). This value can be influenced with [ServerName](#).

### **REQUEST\_URI**

The path component of the requested URI, such as "/index.html". This notably excludes the query string which is available as its own variable named `QUERY_STRING`.

### **THE\_REQUEST**

The full HTTP request line sent by the browser to the server (e.g., "GET /index.html HTTP/1.1"). This does not include any additional headers sent by the browser. This value has not been unescaped (decoded), unlike most other variables below.

If the *TestString* has the special value *expr*, the *CondPattern* will be treated as an [ap\\_expr](#). HTTP headers referenced in the expression will be added to the Vary header if the *novary* flag is not given.

Other things you should be aware of:

1. The variables `SCRIPT_FILENAME` and `REQUEST_FILENAME` contain the same value - the value of the `filename` field of the internal `request_rec` structure of the Apache HTTP Server. The first name is the commonly

known CGI variable name while the second is the appropriate counterpart of REQUEST\_URI (which contains the value of the uri field of request\_rec).

If a substitution occurred and the rewriting continues, the value of both variables will be updated accordingly.

If used in per-server context (*i.e.*, before the request is mapped to the filesystem) SCRIPT\_FILENAME and REQUEST\_FILENAME cannot contain the full local filesystem path since the path is unknown at this stage of processing. Both variables will initially contain the value of REQUEST\_URI in that case. In order to obtain the full local filesystem path of the request in per-server context, use an URL-based look-ahead `%{LA-U:REQUEST_FILENAME}` to determine the final value of REQUEST\_FILENAME.

2. `%{ENV:variable}`, where *variable* can be any environment variable, is also available. This is looked-up via internal Apache httpd structures and (if not found there) via `getenv()` from the Apache httpd server process.
3. `%{SSL:variable}`, where *variable* is the name of an [SSL environment variable](#), can be used whether or not `mod_ssl` is loaded, but will always expand to the empty string if it is not. Example: `%{SSL:SSL_CIPHER_USEKEYSIZE}` may expand to 128. These variables are available even without setting the `StdEnvVars` option of the [SSLOptions](#) directive.
4. `%{HTTP:header}`, where *header* can be any HTTP MIME-header name, can always be used to obtain the value of a header sent in the HTTP request. Example: `%{HTTP:Proxy-Connection}` is the value of the HTTP header `Proxy-Connection:`.

If a HTTP header is used in a condition this header is added

to the Vary header of the response in case the condition evaluates to true for the request. It is **not** added if the condition evaluates to false for the request. Adding the HTTP header to the Vary header of the response is needed for proper caching.

It has to be kept in mind that conditions follow a short circuit logic in the case of the '**ornext |OR**' flag so that certain conditions might not be evaluated at all.

5. `%{LA-U:variable}` can be used for look-aheads which perform an internal (URL-based) sub-request to determine the final value of *variable*. This can be used to access variable for rewriting which is not available at the current stage, but will be set in a later phase.

For instance, to rewrite according to the `REMOTE_USER` variable from within the per-server context (`httpd.conf` file) you must use `%{LA-U:REMOTE_USER}` - this variable is set by the authorization phases, which come *after* the URL translation phase (during which `mod_rewrite` operates).

On the other hand, because `mod_rewrite` implements its per-directory context (`.htaccess` file) via the Fixup phase of the API and because the authorization phases come *before* this phase, you just can use `%{REMOTE_USER}` in that context.

6. `%{LA-F:variable}` can be used to perform an internal (filename-based) sub-request, to determine the final value of *variable*. Most of the time, this is the same as LA-U above.

*CondPattern* is the condition pattern, a regular expression which is applied to the current instance of the *TestString*. *TestString* is first evaluated, before being matched against *CondPattern*.

*CondPattern* is usually a *perl compatible regular expression*, but

there is additional syntax available to perform other useful tests against the *TestString*:

1. You can prefix the pattern string with a '!' character (exclamation mark) to negate the result of the condition, no matter what kind of *CondPattern* is used.
2. You can perform lexicographical string comparisons:

#### **<CondPattern**

Lexicographically precedes

Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* lexicographically precedes *CondPattern*.

#### **>CondPattern**

Lexicographically follows

Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* lexicographically follows *CondPattern*.

#### **=CondPattern**

Lexicographically equal

Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* is lexicographically equal to *CondPattern* (the two strings are exactly equal, character for character). If *CondPattern* is "" (two quotation marks) this compares *TestString* to the empty string.

#### **<=CondPattern**

Lexicographically less than or equal to

Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* lexicographically precedes *CondPattern*, or is equal to *CondPattern* (the two strings are equal, character for character).

## **>=CondPattern**

Lexicographically greater than or equal to  
Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* lexicographically follows *CondPattern*, or is equal to *CondPattern* (the two strings are equal, character for character).

### 3. You can perform integer comparisons:

#### **-eq**

Is numerically **e**qual to  
The *TestString* is treated as an integer, and is numerically compared to the *CondPattern*. True if the two are numerically equal.

#### **-ge**

Is numerically **g**reater than or **e**qual to  
The *TestString* is treated as an integer, and is numerically compared to the *CondPattern*. True if the *TestString* is numerically greater than or equal to the *CondPattern*.

#### **-gt**

Is numerically **g**reater than  
The *TestString* is treated as an integer, and is numerically compared to the *CondPattern*. True if the *TestString* is numerically greater than the *CondPattern*.

#### **-le**

Is numerically **l**ess than or **e**qual to  
The *TestString* is treated as an integer, and is numerically compared to the *CondPattern*. True if the *TestString* is numerically less than or equal to the *CondPattern*. Avoid confusion with the **-l** by using the **-L** or **-h** variant.

#### **-lt**

Is numerically **l**ess than  
The *TestString* is treated as an integer, and is numerically

compared to the *CondPattern*. True if the *TestString* is numerically less than the *CondPattern*. Avoid confusion with the **-l** by using the **-L** or **-h** variant.

**-ne**

Is numerically **not equal** to  
The *TestString* is treated as an integer, and is numerically compared to the *CondPattern*. True if the two are numerically different. This is equivalent to **! -eq**.

4. You can perform various file attribute tests:

**-d**

Is **directory**.  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a directory.

**-f**

Is regular **file**.  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file.

**-F**

Is existing file, via subrequest.  
Checks whether or not *TestString* is a valid file, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!

**-h**

Is symbolic link, bash convention.  
See **-l**.

**-l**

Is symbolic **link**.  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a symbolic link. May also use the

bash convention of **-L** or **-h** if there's a possibility of confusion such as when using the **-lt** or **-le** tests.

**-L**

Is symbolic link, bash convention.  
See **-l**.

**-s**

Is regular file, with **size**.  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file with size greater than zero.

**-U**

Is existing URL, via subrequest.  
Checks whether or not *TestString* is a valid URL, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!

This flag *only* returns information about things like access control, authentication, and authorization. This flag *does not* return information about the status code the configured handler (static file, CGI, proxy, etc.) would have returned.

**-x**

Has executable permissions.  
Treats the *TestString* as a pathname and tests whether or not it exists, and has executable permissions. These permissions are determined according to the underlying OS.

For example:

```
RewriteCond /var/www/%{REQUEST_URI} !-f
```

```
RewriteRule ^(.+) /other/archive/$1 [R]
```

5. If the *TestString* has the special value *expr*, the *CondPattern* will be treated as an [ap\\_expr](#).

In the below example, `-strmatch` is used to compare the `REFERER` against the site hostname, to block unwanted hotlinking.

```
RewriteCond expr "! %{HTTP_REFERER} -str  
RewriteRule "^/images" "-" [F]
```

You can also set special flags for *CondPattern* by appending **[flags]** as the third argument to the `RewriteCond` directive, where *flags* is a comma-separated list of any of the following flags:

- **'nocase | NC'** (no case)  
This makes the test case-insensitive - differences between 'A-Z' and 'a-z' are ignored, both in the expanded *TestString* and the *CondPattern*. This flag is effective only for comparisons between *TestString* and *CondPattern*. It has no effect on filesystem and subrequest checks.
- **'ornext | OR'** (or next condition)  
Use this to combine rule conditions with a local OR instead of the implicit AND. Typical example:

```
RewriteCond "%{REMOTE_HOST}" "^host1"  
RewriteCond "%{REMOTE_HOST}" "^host2"  
RewriteCond "%{REMOTE_HOST}" "^host3"  
RewriteRule ...some special stuff for an
```

Without this flag you would have to write the condition/rule

pair three times.

- **'novary|NV'** (no vary)

If a HTTP header is used in the condition, this flag prevents this header from being added to the Vary header of the response.

Using this flag might break proper caching of the response if the representation of this response varies on the value of this header. So this flag should be only used if the meaning of the Vary header is well understood.

### Example:

To rewrite the Homepage of a site according to the ``User - Agent : " header of the request, you can use the following:

```
RewriteCond  "%{HTTP_USER_AGENT}"  "(iPhone  
RewriteRule  "^/$"                  "/homepag  
  
RewriteRule  "^/$"                  "/homepag
```

Explanation: If you use a browser which identifies itself as a mobile browser (note that the example is incomplete, as there are many other mobile platforms), the mobile version of the homepage is served. Otherwise, the standard page is served.



## RewriteEngine Directive

<b>Description:</b>	Enables or disables runtime rewriting engine
<b>Syntax:</b>	RewriteEngine on off
<b>Default:</b>	RewriteEngine off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteEngine` directive enables or disables the runtime rewriting engine. If it is set to `off` this module does no runtime processing at all. It does not even update the `SCRIPT_URx` environment variables.

Use this directive to disable rules in a particular context, rather than commenting out all the `RewriteRule` directives.

Note that rewrite configurations are not inherited by virtual hosts. This means that you need to have a `RewriteEngine on` directive for each virtual host in which you wish to use rewrite rules.

`RewriteMap` directives of the type `prg` are not started during server initialization if they're defined in a context that does not have `RewriteEngine` set to `on`



<b>Description:</b>	Defines a mapping function for key-lookup
<b>Syntax:</b>	<code>RewriteMap MapName MapType:MapSource</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteMap` directive defines a *Rewriting Map* which can be used inside rule substitution strings by the mapping-functions to insert/substitute fields through a key lookup. The source of this lookup can be of various types.

The *MapName* is the name of the map and will be used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

**`#{ MapName : LookupKey }`**  
**`#{ MapName : LookupKey | DefaultValue }`**

When such a construct occurs, the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the key is not found then it is substituted by *DefaultValue* or by the empty string if no *DefaultValue* was specified. Empty values behave as if the key was absent, therefore it is not possible to distinguish between empty-valued keys and absent keys.

For example, you might define a `RewriteMap` as:

```
RewriteMap examplemap "txt:/path/to/file/map"
```

You would then be able to use this map in a `RewriteRule` as follows:

```
RewriteRule "^/ex/(.*)" "${examplemap:$1}"
```

The following combinations for *MapType* and *MapSource* can be used:

**txt**

A plain text file containing space-separated key-value pairs, one per line. ([Details ...](#))

**rnd**

Randomly selects an entry from a plain text file ([Details ...](#))

**dbm**

Looks up an entry in a dbm file containing name, value pairs. Hash is constructed from a plain text file format using the [httxt2dbm](#) utility. ([Details ...](#))

**int**

One of the four available internal functions provided by RewriteMap: toupper, tolower, escape or unescape. ([Details ...](#))

**prg**

Calls an external program or script to process the rewriting. ([Details ...](#))

**dbd or fastdbd**

A SQL SELECT statement to be performed to look up the rewrite target. ([Details ...](#))

Further details, and numerous examples, may be found in the [RewriteMap HowTo](#)



<b>Description:</b>	Sets some special options for the rewrite engine
<b>Syntax:</b>	RewriteOptions <i>Options</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteOptions` directive sets some special options for the current per-server or per-directory configuration. The *Option* string can currently only be one of the following:

### **Inherit**

This forces the current configuration to inherit the configuration of the parent. In per-virtual-server context, this means that the maps, conditions and rules of the main server are inherited. In per-directory context this means that conditions and rules of the parent directory's `.htaccess` configuration or `<Directory>` sections are inherited. The inherited rules are virtually copied to the section where this directive is being used. If used in combination with local rules, the inherited rules are copied behind the local rules. The position of this directive - below or above of local rules - has no influence on this behavior. If local rules forced the rewriting to stop, the inherited rules won't be processed.

Rules inherited from the parent scope are applied **after** rules specified in the child scope.

### **InheritBefore**

Like `Inherit` above, but the rules from the parent scope are applied **before** rules specified in the child scope.  
Available in Apache HTTP Server 2.3.10 and later.

## **InheritDown**

If this option is enabled, all child configurations will inherit the configuration of the current configuration. It is equivalent to specifying `RewriteOptions Inherit` in all child configurations. See the `Inherit` option for more details on how the parent-child relationships are handled.

Available in Apache HTTP Server 2.4.8 and later.

## **InheritDownBefore**

Like `InheritDown` above, but the rules from the current scope are applied **before** rules specified in any child's scope.

Available in Apache HTTP Server 2.4.8 and later.

## **IgnoreInherit**

This option forces the current and child configurations to ignore all rules that would be inherited from a parent specifying `InheritDown` or `InheritDownBefore`.

Available in Apache HTTP Server 2.4.8 and later.

## **AllowNoSlash**

By default, `mod_rewrite` will ignore URLs that map to a directory on disk but lack a trailing slash, in the expectation that the `mod_dir` module will issue the client with a redirect to the canonical URL with a trailing slash.

When the `DirectorySlash` directive is set to off, the `AllowNoSlash` option can be enabled to ensure that rewrite rules are no longer ignored. This option makes it possible to apply rewrite rules within `.htaccess` files that match the directory without a trailing slash, if so desired.

Available in Apache HTTP Server 2.4.0 and later.

## **AllowAnyURI**

When `RewriteRule` is used in `VirtualHost` or `server` context with version 2.2.22 or later of `httpd`, `mod_rewrite`

will only process the rewrite rules if the request URI is a [URL-path](#). This avoids some security issues where particular rules could allow "surprising" pattern expansions (see [CVE-2011-3368](#) and [CVE-2011-4317](#)). To lift the restriction on matching a URL-path, the `AllowAnyURI` option can be enabled, and `mod_rewrite` will apply the rule set to any request URI string, regardless of whether that string matches the URL-path grammar required by the HTTP specification. Available in Apache HTTP Server 2.4.3 and later.

### Security Warning

Enabling this option will make the server vulnerable to security issues if used with rewrite rules which are not carefully authored. It is **strongly recommended** that this option is not used. In particular, beware of input strings containing the '@' character which could change the interpretation of the transformed URI, as per the above CVE names.

### MergeBase

With this option, the value of `RewriteBase` is copied from where it's explicitly defined into any sub-directory or sub-location that doesn't define its own `RewriteBase`. This was the default behavior in 2.4.0 through 2.4.3, and the flag to restore it is available Apache HTTP Server 2.4.4 and later.

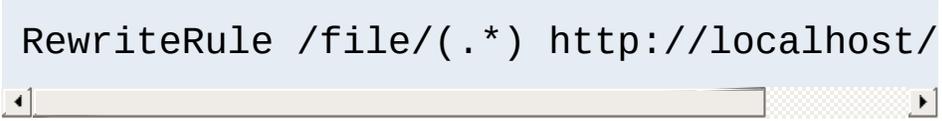
### IgnoreContextInfo

When a relative substitution is made in directory (htaccess) context and `RewriteBase` has not been set, this module uses some extended URL and filesystem context information to change the relative substitution back into a URL. Modules such as `mod_userdir` and `mod_alias` supply this extended context info. Available in 2.4.16 and later.

## LegacyPrefixDocRoot

Prior to 2.4.26, if a substitution was an absolute URL that matched the current virtual host, the URL might first be reduced to a URL-path and then later reduced to a local path. Since the URL can be reduced to a local path, the path should be prefixed with the document root. This prevents a file such as `/tmp/myfile` from being accessed when a request is made to `http://host/file/myfile` with the following [RewriteRule](#).

```
RewriteRule /file/(.*) http://localhost/
```



This option allows the old behavior to be used where the document root is not prefixed to a local path that was reduced from a URL. Available in 2.4.26 and later.



<b>Description:</b>	Defines rules for the rewriting engine
<b>Syntax:</b>	RewriteRule <i>Pattern Substitution</i> [ <i>flags</i> ]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The **RewriteRule** directive is the real rewriting workhorse. The directive can occur more than once, with each instance defining a single rewrite rule. The order in which these rules are defined is important - this is the order in which they will be applied at run-time.

*Pattern* is a perl compatible regular expression. What this pattern is compared against varies depending on where the **RewriteRule** directive is defined.

### What is matched?

- In **VirtualHost** context, The *Pattern* will initially be matched against the part of the URL after the hostname and port, and before the query string (e.g. "/app1/index.html"). This is the (%-decoded) [URL-path](#).
- In per-directory context (**Directory** and .htaccess), the *Pattern* is matched against only a partial path, for example a request of "/app1/index.html" may result in comparison against "app1/index.html" or "index.html" depending on where the **RewriteRule** is defined.

The directory path where the rule is defined is stripped from the currently mapped filesystem path before comparison (up to and including a trailing slash). The net result of this

per-directory prefix stripping is that rules in this context only match against the portion of the currently mapped filesystem path "below" where the rule is defined.

Directives such as `DocumentRoot` and `Alias`, or even the result of previous `RewriteRule` substitutions, determine the currently mapped filesystem path.

- If you wish to match against the hostname, port, or query string, use a `RewriteCond` with the  `%{HTTP_HOST}`,  `%{SERVER_PORT}`, or  `%{QUERY_STRING}` variables respectively.

## Per-directory Rewrites

- The rewrite engine may be used in `.htaccess` files and in `<Directory>` sections, with some additional complexity.
- To enable the rewrite engine in this context, you need to set "RewriteEngine On" **and** "Options FollowSymLinks" must be enabled. If your administrator has disabled override of `FollowSymLinks` for a user's directory, then you cannot use the rewrite engine. This restriction is required for security reasons.
- See the `RewriteBase` directive for more information regarding what prefix will be added back to relative substitutions.
- If you wish to match against the full URL-path in a per-directory (`htaccess`) `RewriteRule`, use the  `%{REQUEST_URI}` variable in a `RewriteCond`.
- The removed prefix always ends with a slash, meaning the matching occurs against a string which *never* has a leading slash. Therefore, a *Pattern* with  `^/` never matches in per-directory context.
- Although rewrite rules are syntactically permitted in

[<Location>](#) and [<Files>](#) sections (including their regular expression counterparts), this should never be necessary and is unsupported. A likely feature to break in these contexts is relative substitutions.

For some hints on [regular expressions](#), see the [mod\\_rewrite Introduction](#).

In `mod_rewrite`, the NOT character ('!') is also available as a possible pattern prefix. This enables you to negate a pattern; to say, for instance: ``if the current URL does **NOT** match this pattern". This can be used for exceptional cases, where it is easier to match the negative pattern, or as a last default rule.

### Note

When using the NOT character to negate a pattern, you cannot include grouped wildcard parts in that pattern. This is because, when the pattern does NOT match (ie, the negation matches), there are no contents for the groups. Thus, if negated patterns are used, you cannot use `$N` in the substitution string!

The *Substitution* of a rewrite rule is the string that replaces the original URL-path that was matched by *Pattern*. The *Substitution* may be a:

### file-system path

Designates the location on the file-system of the resource to be delivered to the client. Substitutions are only treated as a file-system path when the rule is configured in server (virtualhost) context and the first component of the path in the substitution exists in the file-system

### URL-path

A [DocumentRoot](#)-relative path to the resource to be served.

Note that `mod_rewrite` tries to guess whether you have specified a file-system path or a URL-path by checking to see if the first segment of the path exists at the root of the file-system. For example, if you specify a *Substitution* string of `/www/file.html`, then this will be treated as a URL-path *unless* a directory named `www` exists at the root of your file-system (or, in the case of using rewrites in a `.htaccess` file, relative to your document root), in which case it will be treated as a file-system path. If you wish other URL-mapping directives (such as `Alias`) to be applied to the resulting URL-path, use the `[PT]` flag as described below.

### **Absolute URL**

If an absolute URL is specified, `mod_rewrite` checks to see whether the hostname matches the current host. If it does, the scheme and hostname are stripped out and the resulting path is treated as a URL-path. Otherwise, an external redirect is performed for the given URL. To force an external redirect back to the current host, see the `[R]` flag below.

### **- (dash)**

A dash indicates that no substitution should be performed (the existing path is passed through untouched). This is used when a flag (see below) needs to be applied without changing the path.

In addition to plain text, the *Substitution* string can include

1. back-references (`$N`) to the RewriteRule pattern
2. back-references (`%N`) to the last matched RewriteCond pattern
3. server-variables as in rule condition test-strings (`{VARNAME}`)
4. `mapping-function` calls (`${mapname : key | default}`)

Back-references are identifiers of the form  $\$N$  ( $N=0..9$ ), which will be replaced by the contents of the  $N$ th group of the matched *Pattern*. The server-variables are the same as for the *TestString* of a [RewriteCond](#) directive. The mapping-functions come from the [RewriteMap](#) directive and are explained there. These three types of variables are expanded in the order above.

Rewrite rules are applied to the results of previous rewrite rules, in the order in which they are defined in the config file. The URL-path or file-system path (see ["What is matched?"](#), above) is **completely replaced** by the *Substitution* and the rewriting process continues until all rules have been applied, or it is explicitly terminated by an [L flag](#), or other flag which implies immediate termination, such as **END** or **F**.

### Modifying the Query String

By default, the query string is passed through unchanged. You can, however, create URLs in the substitution string containing a query string part. Simply use a question mark inside the substitution string to indicate that the following text should be re-injected into the query string. When you want to erase an existing query string, end the substitution string with just a question mark. To combine new and old query strings, use the [\[QSA\]](#) flag.

Additionally you can set special actions to be performed by appending **[flags]** as the third argument to the [RewriteRule](#) directive. *Flags* is a comma-separated list, surround by square brackets, of any of the flags in the following table. More details, and examples, for each flag, are available in the [Rewrite Flags document](#).

Flag and syntax	Function

B	Escape non-alphanumeric characters in <i>before</i> applying the transformation. <a href="#">details</a>
backrefnoplus BNP	If backreferences are being escaped, space escaped to %20 instead of +. Useful when backreference will be used in the path context than the query string. <a href="#">details ...</a>
chain C	Rule is chained to the following rule. If the rule(s) chained to it will be skipped. <a href="#">details</a>
cookie CO=NAME:VAL	Sets a cookie in the client browser. Full syntax: CO=NAME:VAL:domain[:lifetime[:path[:secure]. <a href="#">details ...</a>
discardpath DPI	Causes the PATH_INFO portion of the request to be discarded. <a href="#">details ...</a>
END	Stop the rewriting process immediately and any more rules. Also prevents further execution of rules in per-directory and .htaccess context (see also the END directive in 2.3.9 and later) <a href="#">details ...</a>
env E=[!]VAR[:VAL]	Causes an environment variable VAR to be set to value VAL if provided). The form !VAR causes environment variable VAR to be unset. <a href="#">details ...</a>
forbidden F	Returns a 403 FORBIDDEN response to the client browser. <a href="#">details ...</a>
gone G	Returns a 410 GONE response to the client browser. <a href="#">details ...</a>
Handler H=Content-handler	Causes the resulting URI to be sent to the Content-handler for processing. <a href="#">details ...</a>
last L	Stop the rewriting process immediately and any more rules. Especially note caveats for per-directory and .htaccess context (see also the END directive in 2.3.9 and later) <a href="#">details ...</a>
next N	Re-run the rewriting process, starting again with the first rule, using the result of the ruleset so far as a point. <a href="#">details ...</a>

<code>nocase NC</code>	Makes the pattern comparison case-insensitive.
<code>noescape NE</code>	Prevent <code>mod_rewrite</code> from applying hexcode escapes to special characters in the result of the rewrite.
<code>nosubreq NS</code>	Causes a rule to be skipped if the current rule is being applied to an internal sub-request. <a href="#">details ...</a>
<code>proxy P</code>	Force the substitution URL to be internally proxied. <a href="#">details ...</a>
<code>passthrough PT</code>	Forces the resulting URI to be passed back to the mapping engine for processing of other URI translators, such as <code>Alias</code> or <code>Redirect</code> .
<code>qsappend QSA</code>	Appends any query string from the original request to any query string created in the rewrite target. <a href="#">details ...</a>
<code>qsdiscard QSD</code>	Discard any query string attached to the incoming request. <a href="#">details ...</a>
<code>qslast QSL</code>	Interpret the last (right-most) question mark as the string delimiter, instead of the first (left-most) one. Available in 2.4.19 and later. <a href="#">details ...</a>
<code>redirect R[=<i>code</i>]</code>	Forces an external redirect, optionally with an HTTP status code. <a href="#">details ...</a>
<code>skip S=<i>num</i></code>	Tells the rewriting engine to skip the next <i>num</i> rules if the current rule matches. <a href="#">details ...</a>
<code>type T=<i>MIME-type</i></code>	Force the <a href="#">MIME-type</a> of the target file to be <i>MIME-type</i> . <a href="#">details ...</a>

## Home directory expansion

When the substitution string begins with a string resembling `"/~user"` (via explicit text or backreferences), `mod_rewrite` performs home directory expansion independent of the presence or configuration of `mod_userdir`.

This expansion does not occur when the `PT` flag is used on the `RewriteRule` directive.

Here are all possible substitution combinations and their meanings:

**Inside per-server configuration (httpd.conf)  
for request "GET /somepath/pathinfo":**

Given Rule	Resulting Substitution
^/somepath(.*) otherpath\$1	invalid, not supported
^/somepath(.*) otherpath\$1 [R]	invalid, not supported
^/somepath(.*) otherpath\$1 [P]	invalid, not supported
^/somepath(.*) /otherpath\$1	/otherpath/pathinfo
^/somepath(.*) /otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^/somepath(.*) /otherpath\$1 [P]	doesn't make sense, not supported
^/somepath(.*) http://thishost/otherpath\$1	/otherpath/pathinfo
^/somepath(.*) http://thishost/otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^/somepath(.*) http://thishost/otherpath\$1 [P]	doesn't make sense, not supported
^/somepath(.*) http://otherhost/otherpath\$1	http://otherhost/otherpath/pathinfo via external redirection
^/somepath(.*) http://otherhost/otherpath\$1 [R]	http://otherhost/otherpath/pathinfo via external redirection (the [R] flag is redundant)
^/somepath(.*)	http://otherhost/otherpath/pathinfo

http://otherhost/otherpath\$1 [P]	via internal proxy
--------------------------------------	--------------------

**Inside per-directory configuration for /somepath  
(/physical/path/to/somepath/.htaccess, with  
RewriteBase "/somepath")  
for request `GET /somepath/localpath/pathinfo`:**

Given Rule	Resulting Substitution
^localpath(*) otherpath\$1	/somepath/otherpath/pathinfo
^localpath(*) otherpath\$1 [R]	http://thishost/somepath/otherpath/pa via external redirection
^localpath(*) otherpath\$1 [P]	doesn't make sense, not supported
^localpath(*) /otherpath\$1	/otherpath/pathinfo
^localpath(*) /otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^localpath(*) /otherpath\$1 [P]	doesn't make sense, not supported
^localpath(*) http://thishost/otherpath\$1	/otherpath/pathinfo
^localpath(*) http://thishost/otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^localpath(*) http://thishost/otherpath\$1 [P]	doesn't make sense, not supported
^localpath(*) http://otherhost/otherpath\$1	http://otherhost/otherpath/pathinfo via external redirection
^localpath(*) http://otherhost/otherpath\$1 [R]	http://otherhost/otherpath/pathinfo via external redirection (the [R] flag is redundant)

<code>^localpath(.*) http://otherhost/otherpath\$1 [P]</code>	<code>http://otherhost/otherpath/pathinfo via internal proxy</code>
---	---

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_sed`

<b>Description:</b>	Filter Input (request) and Output (response) content using sed syntax
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>sed_module</code>
<b>Source File:</b>	<code>mod_sed.c sed0.c sed1.c regexp.c regexp.h sed.h</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

### Summary

`mod_sed` is an in-process content filter. The `mod_sed` filter implements the sed editing commands implemented by the Solaris 10 sed program as described in the [manual page](#). However, unlike sed, `mod_sed` doesn't take data from standard input. Instead, the filter acts on the entity data sent between client and server. `mod_sed` can be used as an input or output filter. `mod_sed` is a content filter, which means that it cannot be used to modify client or server http headers.

The `mod_sed` output filter accepts a chunk of data, executes the sed scripts on the data, and generates the output which is passed to the next filter in the chain.

The `mod_sed` input filter reads the data from the next filter in the chain, executes the sed scripts, and returns the generated data to the caller filter in the filter chain.

Both the input and output filters only process the data if newline characters are seen in the content. At the end of the data, the rest of the data is treated as the last line.

A tutorial article on `mod_sed`, and why it is more powerful than simple string or regular expression search and replace, is available [on the](#)

[author's blog.](#)



### Adding an output filter

```
# In the following example, the sed filter will change the string  
# "monday" to "MON" and the string "sunday" to SUN in html documents  
# before sending to the client.  
<Directory "/var/www/docs/sed">  
    AddOutputFilter Sed html  
    OutputSed "s/monday/MON/g"  
    OutputSed "s/sunday/SUN/g"  
</Directory>
```

### Adding an input filter

```
# In the following example, the sed filter will change the string  
# "monday" to "MON" and the string "sunday" to SUN in the POST data  
# sent to PHP.  
<Directory "/var/www/docs/sed">  
    AddInputFilter Sed php  
    InputSed "s/monday/MON/g"  
    InputSed "s/sunday/SUN/g"  
</Directory>
```



Complete details of the sed command can be found from the [sed manual page](#).

**b**

Branch to the label specified (similar to goto).

**h**

Copy the current line to the hold buffer.

**H**

Append the current line to the hold buffer.

**g**

Copy the hold buffer to the current line.

**G**

Append the hold buffer to the current line.

**x**

Swap the contents of the hold buffer and the current line.



## InputSed Directive

<b>Description:</b>	Sed command to filter request data (typically POST data)
<b>Syntax:</b>	InputSed <i>sed-command</i>
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Experimental
<b>Module:</b>	mod_sed

The **InputSed** directive specifies the sed command to execute on the request data e.g., POST data.



## OutputSed Directive

<b>Description:</b>	Sed command for filtering response content
<b>Syntax:</b>	OutputSed <i>sed-command</i>
<b>Context:</b>	directory, .htaccess
<b>Status:</b>	Experimental
<b>Module:</b>	mod_sed

The **OutputSed** directive specifies the sed command to execute on the response.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_session`

<b>Description:</b>	Session support
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>session_module</code>
<b>Source File:</b>	<code>mod_session.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

### Warning

The session modules make use of HTTP cookies, and as such can fall victim to Cross Site Scripting attacks, or expose potentially private information to clients. Please ensure that the relevant risks have been taken into account before enabling the session functionality on your server.

This module provides support for a server wide per user session interface. Sessions can be used for keeping track of whether a user has been logged in, or for other per user information that should be kept available across requests.

Sessions may be stored on the server, or may be stored on the browser. Sessions may also be optionally encrypted for added security. These features are divided into several modules in addition to `mod_session`; `mod_session_crypto`, `mod_session_cookie` and `mod_session_dbd`. Depending on the server requirements, load the appropriate modules into the server (either statically at compile time or dynamically via the `LoadModule` directive).

Sessions may be manipulated from other modules that depend on the session, or the session may be read from and written to using

environment variables and HTTP headers, as appropriate.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_session\\_cookie](#)

[mod\\_session\\_crypto](#)

[mod\\_session\\_dbd](#)



---

At the core of the session interface is a table of key and value pairs that are made accessible across browser requests. These pairs can be set to any valid string, as needed by the application making use of the session.

The "session" is a **application/x-www-form-urlencoded** string containing these key value pairs, as defined by the [HTML specification](#).

The session can optionally be encrypted and base64 encoded before being written to the storage mechanism, as defined by the administrator.



---

The session interface is primarily developed for the use by other server modules, such as [mod\\_auth\\_form](#), however CGI based applications can optionally be granted access to the contents of the session via the HTTP\_SESSION environment variable. Sessions have the option to be modified and/or updated by inserting an HTTP response header containing the new session parameters.



## Keeping Sessions on the Server

Apache can be configured to keep track of per user sessions stored on a particular server or group of servers. This functionality is similar to the sessions available in typical application servers.

If configured, sessions are tracked through the use of a session ID that is stored inside a cookie, or extracted from the parameters embedded within the URL query string, as found in a typical GET request.

As the contents of the session are stored exclusively on the server, there is an expectation of privacy of the contents of the session. This does have performance and resource implications should a large number of sessions be present, or where a large number of web servers have to share sessions with one another.

The [mod\\_session\\_dbd](#) module allows the storage of user sessions within a SQL database via [mod\\_dbd](#).



## Keeping sessions on the browser

In high traffic environments where keeping track of a session on a server is too resource intensive or inconvenient, the option exists to store the contents of the session within a cookie on the client browser instead.

This has the advantage that minimal resources are required on the server to keep track of sessions, and multiple servers within a server farm have no need to share session information.

The contents of the session however are exposed to the client, with a corresponding risk of a loss of privacy. The `mod_session_crypto` module can be configured to encrypt the contents of the session before writing the session to the client.

The `mod_session_cookie` allows the storage of user sessions on the browser within an HTTP cookie.



Creating a session is as simple as turning the session on, and deciding where the session will be stored. In this example, the session will be stored on the browser, in a cookie called `session`.

### Browser based session

```
Session On
SessionCookieName session path=/
```

The session is not useful unless it can be written to or read from. The following example shows how values can be injected into the session through the use of a predetermined HTTP response header called `X-Replace-Session`.

### Writing to a session

```
Session On
SessionCookieName session path=/
SessionHeader X-Replace-Session
```

The header should contain name value pairs expressed in the same format as a query string in a URL, as in the example below. Setting a key to the empty string has the effect of removing that key from the session.

### CGI to write to a session

```
#!/bin/bash
echo "Content-Type: text/plain"
echo "X-Replace-Session: key1=foo&key2=&key3=bar"
echo
env
```

If configured, the session can be read back from the `HTTP_SESSION` environment variable. By default, the session is kept private, so this has to be explicitly turned on with the [SessionEnv](#) directive.

## Read from a session

```
Session On  
SessionEnv On  
SessionCookieName session path=/  
SessionHeader X-Replace-Session
```

Once read, the CGI variable HTTP\_SESSION should contain the value key1=foo&key3=bar.



Using the "show cookies" feature of your browser, you would have seen a clear text representation of the session. This could potentially be a problem should the end user need to be kept unaware of the contents of the session, or where a third party could gain unauthorised access to the data within the session.

The contents of the session can be optionally encrypted before being placed on the browser using the [mod\\_session\\_crypto](#) module.

### Browser based encrypted session

```
Session On  
SessionCryptoPassphrase secret  
SessionCookieName session path=/  

```

The session will be automatically decrypted on load, and encrypted on save by Apache, the underlying application using the session need have no knowledge that encryption is taking place.

Sessions stored on the server rather than on the browser can also be encrypted as needed, offering privacy where potentially sensitive information is being shared between web servers in a server farm using the [mod\\_session\\_dbd](#) module.



The HTTP cookie mechanism also offers privacy features, such as the ability to restrict cookie transport to SSL protected pages only, or to prevent browser based javascript from gaining access to the contents of the cookie.

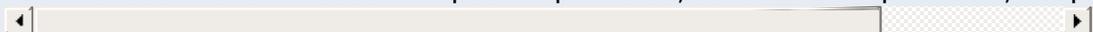
### Warning

Some of the HTTP cookie privacy features are either non-standard, or are not implemented consistently across browsers. The session modules allow you to set cookie parameters, but it makes no guarantee that privacy will be respected by the browser. If security is a concern, use the [mod\\_session\\_crypto](#) to encrypt the contents of the session, or store the session on the server using the [mod\\_session\\_dbd](#) module.

Standard cookie parameters can be specified after the name of the cookie, as in the example below.

### Setting cookie parameters

```
Session On
SessionCryptoPassphrase secret
SessionCookieName session path=/private;domain=example.com;http
```



In cases where the Apache server forms the frontend for backend origin servers, it is possible to have the session cookies removed from the incoming HTTP headers using the [SessionCookieRemove](#) directive. This keeps the contents of the session cookies from becoming accessible from the backend server.



As is possible within many application servers, authentication modules can use a session for storing the username and password after login. The [mod\\_auth\\_form](#) saves the user's login name and password within the session.

### Form based authentication

```
Session On
SessionCryptoPassphrase secret
SessionCookieName session path=/
AuthFormProvider file
AuthUserFile "conf/passwd"
AuthType form
AuthName realm
#...
```

See the [mod\\_auth\\_form](#) module for documentation and complete examples.



In order for sessions to be useful, it must be possible to share the contents of a session with external applications, and it must be possible for an external application to write a session of its own.

A typical example might be an application that changes a user's password set by [mod\\_auth\\_form](#). This application would need to read the current username and password from the session, make the required changes to the user's password, and then write the new password to the session in order to provide a seamless transition to the new password.

A second example might involve an application that registers a new user for the first time. When registration is complete, the username and password is written to the session, providing a seamless transition to being logged in.

### Apache modules

Modules within the server that need access to the session can use the **mod\_session.h** API in order to read from and write to the session. This mechanism is used by modules like [mod\\_auth\\_form](#).

### CGI programs and scripting languages

Applications that run within the webserver can optionally retrieve the value of the session from the **HTTP\_SESSION** environment variable. The session should be encoded as a **application/x-www-form-urlencoded** string as described by the [HTML specification](#). The environment variable is controlled by the setting of the [SessionEnv](#) directive. The session can be written to by the script by returning a **application/x-www-form-urlencoded** response header with a name set by the [SessionHeader](#) directive. In both cases, any encryption or decryption, and the reading the session from or writing the session to the chosen storage mechanism

is handled by the [mod\\_session](#) modules and corresponding configuration.

### Applications behind [mod\\_proxy](#)

If the [SessionHeader](#) directive is used to define an HTTP request header, the session, encoded as a **application/x-www-form-urlencoded** string, will be made available to the application. If the same header is provided in the response, the value of this response header will be used to replace the session. As above, any encryption or decryption, and the reading the session from or writing the session to the chosen storage mechanism is handled by the [mod\\_session](#) modules and corresponding configuration.

### Standalone applications

Applications might choose to manipulate the session outside the control of the Apache HTTP server. In this case, it is the responsibility of the application to read the session from the chosen storage mechanism, decrypt the session, update the session, encrypt the session and write the session to the chosen storage mechanism, as appropriate.



<b>Description:</b>	Enables a session for the current directory or location
<b>Syntax:</b>	Session On Off
<b>Default:</b>	Session Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_session

The **Session** directive enables a session for the directory or location container. Further directives control where the session will be stored and how privacy is maintained.



**Description:** Control whether the contents of the session are written to the *HTTP\_SESSION* environment variable

**Syntax:** `SessionEnv On|Off`

**Default:** `SessionEnv Off`

**Context:** server config, virtual host, directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** `mod_session`

If set to *On*, the `SessionEnv` directive causes the contents of the session to be written to a CGI environment variable called *HTTP\_SESSION*.

The string is written in the URL query format, for example:

```
key1=foo&key3=bar
```



<b>Description:</b>	Define URL prefixes for which a session is ignored
<b>Syntax:</b>	<code>SessionExclude path</code>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session

The `SessionExclude` directive allows sessions to be disabled relative to URL prefixes only. This can be used to make a website more efficient, by targeting a more precise URL space for which a session should be maintained. By default, all URLs within the directory or location are included in the session. The `SessionExclude` directive takes precedence over the `SessionInclude` directive.

### Warning

This directive has a similar purpose to the `path` attribute in HTTP cookies, but should not be confused with this attribute. This directive does not set the `path` attribute, which must be configured separately.



<b>Description:</b>	Import session updates from a given HTTP response header
<b>Syntax:</b>	SessionHeader <i>header</i>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_session

The `SessionHeader` directive defines the name of an HTTP response header which, if present, will be parsed and written to the current session.

The header value is expected to be in the URL query format, for example:

```
key1=foo&key2=&key3=bar
```

Where a key is set to the empty string, that key will be removed from the session.



<b>Description:</b>	Define URL prefixes for which a session is valid
<b>Syntax:</b>	SessionInclude <i>path</i>
<b>Default:</b>	all URLs
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_session

The `SessionInclude` directive allows sessions to be made valid for specific URL prefixes only. This can be used to make a website more efficient, by targeting a more precise URL space for which a session should be maintained. By default, all URLs within the directory or location are included in the session.

### Warning

This directive has a similar purpose to the `path` attribute in HTTP cookies, but should not be confused with this attribute. This directive does not set the `path` attribute, which must be configured separately.



<b>Description:</b>	Define a maximum age in seconds for a session
<b>Syntax:</b>	<code>SessionMaxAge <i>maxage</i></code>
<b>Default:</b>	<code>SessionMaxAge 0</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_session

The `SessionMaxAge` directive defines a time limit for which a session will remain valid. When a session is saved, this time limit is reset and an existing session can be continued. If a session becomes older than this limit without a request to the server to refresh the session, the session will time out and be removed. Where a session is used to stored user login details, this has the effect of logging the user out automatically after the given time.

Setting the maxage to zero disables session expiry.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_session_cookie`

<b>Description:</b>	Cookie based session support
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>session_cookie_module</code>
<b>Source File:</b>	<code>mod_session_cookie.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

### Warning

The session modules make use of HTTP cookies, and as such can fall victim to Cross Site Scripting attacks, or expose potentially private information to clients. Please ensure that the relevant risks have been taken into account before enabling the session functionality on your server.

This submodule of `mod_session` provides support for the storage of user sessions on the remote browser within HTTP cookies.

Using cookies to store a session removes the need for the server or a group of servers to store the session locally, or collaborate to share a session, and can be useful for high traffic environments where a server based session might be too resource intensive.

If session privacy is required, the `mod_session_crypto` module can be used to encrypt the contents of the session before writing the session to the client.

For more details on the session interface, see the documentation for the `mod_session` module.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_session](#)

[mod\\_session\\_crypto](#)

[mod\\_session\\_dbd](#)



## Basic Examples

To create a simple session and store it in a cookie called *session*, configure the session as follows:

### Browser based session

```
Session On  
SessionCookieName session path=/  

```

For more examples on how the session can be configured to be read from and written to by a CGI application, see the [mod\\_session](#) examples section.

For documentation on how the session can be used to store username and password details, see the [mod\\_auth\\_form](#) module.



<b>Description:</b>	Name and attributes for the RFC2109 cookie storing the session
<b>Syntax:</b>	<code>SessionCookieName <i>name attributes</i></code>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_cookie

The `SessionCookieName` directive specifies the name and optional attributes of an RFC2109 compliant cookie inside which the session will be stored. RFC2109 cookies are set using the Set-Cookie HTTP header.

An optional list of cookie attributes can be specified, as per the example below. These attributes are inserted into the cookie as is, and are not interpreted by Apache. Ensure that your attributes are defined correctly as per the cookie specification.

### Cookie with attributes

```
Session On
SessionCookieName session path=/private;domain=example.com;http
```



<b>Description:</b>	Name and attributes for the RFC2965 cookie storing the session
<b>Syntax:</b>	<code>SessionCookieName2 name attributes</code>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_cookie

The `SessionCookieName2` directive specifies the name and optional attributes of an RFC2965 compliant cookie inside which the session will be stored. RFC2965 cookies are set using the `Set-Cookie2` HTTP header.

An optional list of cookie attributes can be specified, as per the example below. These attributes are inserted into the cookie as is, and are not interpreted by Apache. Ensure that your attributes are defined correctly as per the cookie specification.

### Cookie2 with attributes

```
Session On
SessionCookieName2 session path=/private;domain=example.com;htt
```



<b>Description:</b>	Control for whether session cookies should be removed from incoming HTTP headers
<b>Syntax:</b>	SessionCookieRemove On Off
<b>Default:</b>	SessionCookieRemove Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_cookie

The `SessionCookieRemove` flag controls whether the cookies containing the session will be removed from the headers during request processing.

In a reverse proxy situation where the Apache server acts as a server frontend for a backend origin server, revealing the contents of the session cookie to the backend could be a potential privacy violation. When set to on, the session cookie will be removed from the incoming HTTP headers.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_session_crypto`

<b>Description:</b>	Session encryption support
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>session_crypto_module</code>
<b>Source File:</b>	<code>mod_session_crypto.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

### Warning

The session modules make use of HTTP cookies, and as such can fall victim to Cross Site Scripting attacks, or expose potentially private information to clients. Please ensure that the relevant risks have been taken into account before enabling the session functionality on your server.

This submodule of `mod_session` provides support for the encryption of user sessions before being written to a local database, or written to a remote browser via an HTTP cookie.

This can help provide privacy to user sessions where the contents of the session should be kept private from the user, or where protection is needed against the effects of cross site scripting attacks.

For more details on the session interface, see the documentation for the `mod_session` module.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_session](#)

[mod\\_session\\_cookie](#)

[mod\\_session\\_dbd](#)



To create a simple encrypted session and store it in a cookie called *session*, configure the session as follows:

### Browser based encrypted session

```
Session On  
SessionCookieName session path=/  
SessionCryptoPassphrase secret
```

The session will be encrypted with the given key. Different servers can be configured to share sessions by ensuring the same encryption key is used on each server.

If the encryption key is changed, sessions will be invalidated automatically.

For documentation on how the session can be used to store username and password details, see the [mod\\_auth\\_form](#) module.



## SessionCryptoCipher Directive

<b>Description:</b>	The crypto cipher to be used to encrypt the session
<b>Syntax:</b>	SessionCryptoCipher <i>name</i>
<b>Default:</b>	aes256
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Experimental
<b>Module:</b>	mod_session_crypto
<b>Compatibility:</b>	Available in Apache 2.3.0 and later

The `SessionCryptoCipher` directive allows the cipher to be used during encryption. If not specified, the cipher defaults to aes256.

Possible values depend on the crypto driver in use, and could be one of:

- 3des192
- aes128
- aes192
- aes256



**Description:** The crypto driver to be used to encrypt the session

**Syntax:** `SessionCryptoDriver name [param[=value]]`

**Default:** none

**Context:** server config

**Status:** Experimental

**Module:** mod\_session\_crypto

**Compatibility:** Available in Apache 2.3.0 and later

The `SessionCryptoDriver` directive specifies the name of the crypto driver to be used for encryption. If not specified, the driver defaults to the recommended driver compiled into APR-util.

The NSS crypto driver requires some parameters for configuration, which are specified as parameters with optional values after the driver name.

### NSS without a certificate database

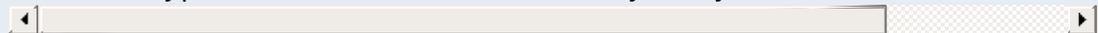
```
SessionCryptoDriver nss
```

### NSS with certificate database

```
SessionCryptoDriver nss dir=certs
```

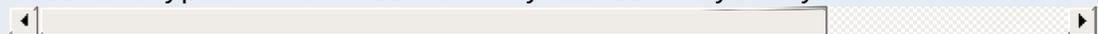
### NSS with certificate database and parameters

```
SessionCryptoDriver nss dir=certs key3=key3.db cert7=cert7.db se
```



### NSS with paths containing spaces

```
SessionCryptoDriver nss "dir=My Certs" key3=key3.db cert7=cert7
```



The *NSS* crypto driver might have already been configured by another part of the server, for example from `mod_nss` or `mod_ldap`. If found to have already been configured, a warning will be logged, and the existing configuration will have taken affect. To avoid this warning, use the `noinit` parameter as follows.

### **NSS with certificate database**

```
SessionCryptoDriver nss noinit
```

To prevent confusion, ensure that all modules requiring *NSS* are configured with identical parameters.

The *openssl* crypto driver supports an optional parameter to specify the engine to be used for encryption.

### **OpenSSL with engine support**

```
SessionCryptoDriver openssl engine=name
```



<b>Description:</b>	The key used to encrypt the session
<b>Syntax:</b>	<code>SessionCryptoPassphrase <i>secret</i> [<i>secret</i> ... ]</code>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Experimental
<b>Module:</b>	mod_session_crypto
<b>Compatibility:</b>	Available in Apache 2.3.0 and later

The `SessionCryptoPassphrase` directive specifies the keys to be used to enable symmetrical encryption on the contents of the session before writing the session, or decrypting the contents of the session after reading the session.

Keys are more secure when they are long, and consist of truly random characters. Changing the key on a server has the effect of invalidating all existing sessions.

Multiple keys can be specified in order to support key rotation. The first key listed will be used for encryption, while all keys listed will be attempted for decryption. To rotate keys across multiple servers over a period of time, add a new secret to the end of the list, and once rolled out completely to all servers, remove the first key from the start of the list.

As of version 2.4.7 if the value begins with `exec:` the resulting command will be executed and the first line returned to standard output by the program will be used as the key.

```
#key used as-is
SessionCryptoPassphrase secret

#Run /path/to/program to get key
SessionCryptoPassphrase exec:/path/to/program
```

```
#Run /path/to/otherProgram and provide arguments  
SessionCryptoPassphrase "exec:/path/to/otherProgram argument1"
```



<b>Description:</b>	File containing keys used to encrypt the session
<b>Syntax:</b>	SessionCryptoPassphraseFile <i>filename</i>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Experimental
<b>Module:</b>	mod_session_crypto
<b>Compatibility:</b>	Available in Apache 2.3.0 and later

The `SessionCryptoPassphraseFile` directive specifies the name of a configuration file containing the keys to use for encrypting or decrypting the session, specified one per line. The file is read on server start, and a graceful restart will be necessary for httpd to pick up changes to the keys.

Unlike the `SessionCryptoPassphrase` directive, the keys are not exposed within the httpd configuration and can be hidden by protecting the file appropriately.

Multiple keys can be specified in order to support key rotation. The first key listed will be used for encryption, while all keys listed will be attempted for decryption. To rotate keys across multiple servers over a period of time, add a new secret to the end of the list, and once rolled out completely to all servers, remove the first key from the start of the list.



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_session_dbd`

<b>Description:</b>	DBD/SQL based session support
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>session_dbd_module</code>
<b>Source File:</b>	<code>mod_session_dbd.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

### Summary

#### Warning

The session modules make use of HTTP cookies, and as such can fall victim to Cross Site Scripting attacks, or expose potentially private information to clients. Please ensure that the relevant risks have been taken into account before enabling the session functionality on your server.

This submodule of `mod_session` provides support for the storage of user sessions within a SQL database using the `mod_dbd` module.

Sessions can either be **anonymous**, where the session is keyed by a unique UUID string stored on the browser in a cookie, or **per user**, where the session is keyed against the userid of the logged in user.

SQL based sessions are hidden from the browser, and so offer a measure of privacy without the need for encryption.

Different webservers within a server farm may choose to share a database, and so share sessions with one another.

For more details on the session interface, see the documentation for the `mod_session` module.



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[mod\\_session](#)

[mod\\_session\\_crypto](#)

[mod\\_session\\_cookie](#)

[mod\\_dbd](#)



Before the `mod_session_dbd` module can be configured to maintain a session, the `mod_dbd` module must be configured to make the various database queries available to the server.

There are four queries required to keep a session maintained, to select an existing session, to update an existing session, to insert a new session, and to delete an expired or empty session. These queries are configured as per the example below.

### Sample DBD configuration

```
DBDriver pgsq1
DBDParams "dbname=apachesession user=apache password=xxxxx host:
DBDPrepareSQL "delete from session where key = %s" deletesession
DBDPrepareSQL "update session set value = %s, expiry = %lld, key
DBDPrepareSQL "insert into session (value, expiry, key) values
DBDPrepareSQL "select value from session where key = %s and (ex
DBDPrepareSQL "delete from session where expiry != 0 and expiry
```



Anonymous sessions are keyed against a unique UUID, and stored on the browser within an HTTP cookie. This method is similar to that used by most application servers to store session information.

To create a simple anonymous session and store it in a postgres database table called *apachesession*, and save the session ID in a cookie called *session*, configure the session as follows:

### SQL based anonymous session

```
Session On  
SessionDBDCookieName session path=/  

```

For more examples on how the session can be configured to be read from and written to by a CGI application, see the [mod\\_session](#) examples section.

For documentation on how the session can be used to store username and password details, see the [mod\\_auth\\_form](#) module.



Per user sessions are keyed against the username of a successfully authenticated user. It offers the most privacy, as no external handle to the session exists outside of the authenticated realm.

Per user sessions work within a correctly configured authenticated environment, be that using basic authentication, digest authentication or SSL client certificates. Due to the limitations of who came first, the chicken or the egg, per user sessions cannot be used to store authentication credentials from a module like [mod\\_auth\\_form](#).

To create a simple per user session and store it in a postgres database table called *apachesession*, and with the session keyed to the userid, configure the session as follows:

### SQL based per user session

```
Session On  
SessionDBDPerUser On
```



Over the course of time, the database can be expected to start accumulating expired sessions. At this point, the `mod_session_dbd` module is not yet able to handle session expiry automatically.

### **Warning**

The administrator will need to set up an external process via cron to clean out expired sessions.



<b>Description:</b>	Name and attributes for the RFC2109 cookie storing the session ID
<b>Syntax:</b>	<code>SessionDBDCookieName <i>name attributes</i></code>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_dbd

The `SessionDBDCookieName` directive specifies the name and optional attributes of an RFC2109 compliant cookie inside which the session ID will be stored. RFC2109 cookies are set using the Set-Cookie HTTP header.

An optional list of cookie attributes can be specified, as per the example below. These attributes are inserted into the cookie as is, and are not interpreted by Apache. Ensure that your attributes are defined correctly as per the cookie specification.

### Cookie with attributes

```
Session On
SessionDBDCookieName session path=/private;domain=example.com;htt
```



<b>Description:</b>	Name and attributes for the RFC2965 cookie storing the session ID
<b>Syntax:</b>	<code>SessionDBDCookieName2 name attributes</code>
<b>Default:</b>	none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_dbd

The `SessionDBDCookieName2` directive specifies the name and optional attributes of an RFC2965 compliant cookie inside which the session ID will be stored. RFC2965 cookies are set using the Set-Cookie2 HTTP header.

An optional list of cookie attributes can be specified, as per the example below. These attributes are inserted into the cookie as is, and are not interpreted by Apache. Ensure that your attributes are defined correctly as per the cookie specification.

### Cookie2 with attributes

```
Session On
SessionDBDCookieName2 session path=/private;domain=example.com;|
```



<b>Description:</b>	Control for whether session ID cookies should be removed from incoming HTTP headers
<b>Syntax:</b>	SessionDBDCookieRemove On Off
<b>Default:</b>	SessionDBDCookieRemove On
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_dbd

The `SessionDBDCookieRemove` flag controls whether the cookies containing the session ID will be removed from the headers during request processing.

In a reverse proxy situation where the Apache server acts as a server frontend for a backend origin server, revealing the contents of the session ID cookie to the backend could be a potential privacy violation. When set to on, the session ID cookie will be removed from the incoming HTTP headers.



<b>Description:</b>	The SQL query to use to remove sessions from the database
<b>Syntax:</b>	<code>SessionDBDeleteLabel <i>label</i></code>
<b>Default:</b>	<code>SessionDBDeleteLabel deletesession</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_session_dbd</code>

The `SessionDBDeleteLabel` directive sets the default delete query label to be used to delete an expired or empty session. This label must have been previously defined using the `DBDPrepareSQL` directive.



**Description:** The SQL query to use to insert sessions into the database

**Syntax:** `SessionDBDInsertLabel label`

**Default:** `SessionDBDInsertLabel insertsession`

**Context:** server config, virtual host, directory, .htaccess

**Status:** Extension

**Module:** `mod_session_dbd`

The `SessionDBDInsertLabel` directive sets the default insert query label to be used to load in a session. This label must have been previously defined using the `DBDPrepareSQL` directive.

If an attempt to update the session affects no rows, this query will be called to insert the session into the database.



<b>Description:</b>	Enable a per user session
<b>Syntax:</b>	SessionDBDPerUser On Off
<b>Default:</b>	SessionDBDPerUser Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	mod_session_dbd

The `SessionDBDPerUser` flag enables a per user session keyed against the user's login name. If the user is not logged in, this directive will be ignored.



<b>Description:</b>	The SQL query to use to select sessions from the database
<b>Syntax:</b>	<code>SessionDBDSelectLabel <i>label</i></code>
<b>Default:</b>	<code>SessionDBDSelectLabel selectsession</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_session_dbd</code>

The `SessionDBDSelectLabel` directive sets the default select query label to be used to load in a session. This label must have been previously defined using the `DBDPrepareSQL` directive.



<b>Description:</b>	The SQL query to use to update existing sessions in the database
<b>Syntax:</b>	<code>SessionDBDUpdateLabel <i>label</i></code>
<b>Default:</b>	<code>SessionDBDUpdateLabel updatesession</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_session_dbd</code>

The `SessionDBDUpdateLabel` directive sets the default update query label to be used to load in a session. This label must have been previously defined using the `DBDPrepareSQL` directive.

If an attempt to update the session affects no rows, the insert query will be called to insert the session into the database. If the database supports `InsertOrUpdate`, override this query to perform the update in one query instead of two.



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_setenvif

```
┆  
┆  
┆ Base  
┆ setenvif_module  
┆ mod_setenvif.c
```

## mod\_setenvif

MSIE mozilla

```
BrowserMatch ^Mozilla netscape  
BrowserMatch MSIE !netscape
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)





## BROWSERMATCH

```
: HTTP User-Agent
: BrowserMatch regex [!]env-
: variable[=value] [[!]env-
: variable[=value]] ...
: , , directory, .htaccess
Override : FileInfo
: Base
: mod_setenvif
```

BrowserMatch SetEnvIf , HTTP Use  
Agent . :

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```

:

```
BrowserMatch ^Mozilla forms jpeg=yes browser=netscape
BrowserMatch "^Mozilla/[2-3]" tables agif frames javascript
BrowserMatch MSIE !javascript
```



```

: User-Agent
: BrowserMatchNoCase regex [!]env-
variable[=value] [[!]env-
variable[=value]] ...
: , , directory, .htaccess
Override : FileInfo
: Base
: mod_setenvif

```

BrowserMatchNoCase BrowserMatch .

```

BrowserMatchNoCase mac platform=macintosh
BrowserMatchNoCase win platform=windows

```

BrowserMatch BrowserMatchNoCase SetEnvIf  
SetEnvIfNoCase . :

```

BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot

```



```

:
:
:      SetEnvIf attribute regex [!]env-
:      variable[=value] [[!]env-
:      variable[=value]] ...
:      , , directory, .htaccess
Override : FileInfo
:      Base
:      mod_setenvif

```

**SetEnvIf** . *attri*

1. HTTP ( [RFC2616](#) ); : Host, User - Agent, Referer, Accept - Language.

2. :
- Remote\_Host - ()
  - Remote\_Addr - IP
  - Server\_Addr - IP (2.0.43 )
  - Request\_Method - (GET, POST, )
  - Request\_Protocol - ( , "HTTP/0 "HTTP/1.1",. )
  - Request\_URI - HTTP -- URL (scheme)

3. . **SetEnvIf** .  
 SetEnvIf[NoCase] . " ( )

( regex) [Perl](#) . POSIX.2 egrep . regex



- [:~](#) Sets environment variables based on an `ap_expr` expression
- [:~](#)
- [:~](#) , , directory, .htaccess
- [:~](#) Base
- [:~](#) `mod_setenvif`

Documentation not yet translated. Please see English version of document.



```

:
:      SetEnvIfNoCase attribute regex [!]env-
:      variable[=value] [[!]env-
:      variable[=value]] ...
:      , , directory, .htaccess
Override : FileInfo
:      Base
:      mod_setenvif

```

```

SetEnvIfNoCase      SetEnvIf ,      .
:

```

```

SetEnvIfNoCase Host Apache\.Org site=apache

```

```

HTTP      Host: Apache.Org, apache.org
site " apache".

```



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_slotmem_plain`

<b>Description:</b>	Slot-based shared memory provider.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>slotmem_plain_module</code>
<b>Source File:</b>	<code>mod_slotmem_plain.c</code>

## Summary

`mod_slotmem_plain` is a memory provider which provides for creation and access to a plain memory segment in which the datasets are organized in "slots."

If the memory needs to be shared between threads and processes, a better provider would be [mod\\_slotmem\\_shm](#).

`mod_slotmem_plain` provides the following API functions:

**`apr_status_t doall(ap_slotmem_instance_t *s, ap_slotmem_callback_fn_t *func, void *data, apr_pool_t *pool)`**  
call the callback on all worker slots

**`apr_status_t create(ap_slotmem_instance_t **new, const char *name, apr_size_t item_size, unsigned int item_num, ap_slotmem_type_t type, apr_pool_t *pool)`**  
create a new slotmem with each item size is `item_size`.

**`apr_status_t attach(ap_slotmem_instance_t **new, const char *name, apr_size_t *item_size, unsigned int *item_num, apr_pool_t *pool)`**  
attach to an existing slotmem.

**`apr_status_t dptr(ap_slotmem_instance_t *s, unsigned int item_id, void**mem)`**  
get the direct pointer to the memory associated with this worker slot.

**apr\_status\_t get(ap\_slotmem\_instance\_t \*s, unsigned int item\_id, unsigned char \*dest, apr\_size\_t dest\_len)**

get/read the memory from this slot to dest

**apr\_status\_t put(ap\_slotmem\_instance\_t \*slot, unsigned int item\_id, unsigned char \*src, apr\_size\_t src\_len)**

put/write the data from src to this slot

**unsigned int num\_slots(ap\_slotmem\_instance\_t \*s)**

return the total number of slots in the segment

**apr\_size\_t slot\_size(ap\_slotmem\_instance\_t \*s)**

return the total data size, in bytes, of a slot in the segment

**apr\_status\_t grab(ap\_slotmem\_instance\_t \*s, unsigned int \*item\_id);**

grab or allocate the first free slot and mark as in-use (does not do any data copying)

**apr\_status\_t fgrab(ap\_slotmem\_instance\_t \*s, unsigned int item\_id);**

forced grab or allocate the specified slot and mark as in-use (does not do any data copying)

**apr\_status\_t release(ap\_slotmem\_instance\_t \*s, unsigned int item\_id);**

release or free a slot and mark as not in-use (does not do any data copying)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_slotmem_shm`

<b>Description:</b>	Slot-based shared memory provider.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>slotmem_shm_module</code>
<b>Source File:</b>	<code>mod_slotmem_shm.c</code>

## Summary

`mod_slotmem_shm` is a memory provider which provides for creation and access to a shared memory segment in which the datasets are organized in "slots."

All shared memory is cleared and cleaned with each restart, whether graceful or not. The data itself is stored and restored within a file noted by the `name` parameter in the `create` and `attach` calls. If not specified with an absolute path, the file will be created relative to the path specified by the `DefaultRuntimeDir` directive.

`mod_slotmem_shm` provides the following API functions:

**`apr_status_t doall(ap_slotmem_instance_t *s,`  
**`ap_slotmem_callback_fn_t *func, void *data, apr_pool_t *pool)`**  
call the callback on all worker slots**

**`apr_status_t create(ap_slotmem_instance_t **new, const char`  
**`*name, apr_size_t item_size, unsigned int item_num,`  
**`ap_slotmem_type_t type, apr_pool_t *pool)`**  
create a new slotmem with each item size is `item_size`. `name` is used to generate a filename for the persistent store of the shared memory if configured. Values are:****

**"none"**

Anonymous shared memory and no persistent store

**"file-name"**

[DefaultRuntimeDir]/file-name

**"/absolute-file-name"**

Absolute file name

**apr\_status\_t attach(ap\_slotmem\_instance\_t \*\*new, const char \*name, apr\_size\_t \*item\_size, unsigned int \*item\_num, apr\_pool\_t \*pool)**

attach to an existing slotmem. See create for description of name parameter.

**apr\_status\_t dptr(ap\_slotmem\_instance\_t \*s, unsigned int item\_id, void\*\*mem)**

get the direct pointer to the memory associated with this worker slot.

**apr\_status\_t get(ap\_slotmem\_instance\_t \*s, unsigned int item\_id, unsigned char \*dest, apr\_size\_t dest\_len)**

get/read the memory from this slot to dest

**apr\_status\_t put(ap\_slotmem\_instance\_t \*slot, unsigned int item\_id, unsigned char \*src, apr\_size\_t src\_len)**

put/write the data from src to this slot

**unsigned int num\_slots(ap\_slotmem\_instance\_t \*s)**

return the total number of slots in the segment

**apr\_size\_t slot\_size(ap\_slotmem\_instance\_t \*s)**

return the total data size, in bytes, of a slot in the segment

**apr\_status\_t grab(ap\_slotmem\_instance\_t \*s, unsigned int \*item\_id);**

grab or allocate the first free slot and mark as in-use (does not do any data copying)

**apr\_status\_t fgrab(ap\_slotmem\_instance\_t \*s, unsigned int item\_id);**

forced grab or allocate the specified slot and mark as in-use (does not do any data copying)

**apr\_status\_t** **release**(**ap\_slotmem\_instance\_t** \*s, unsigned int **item\_id**);

release or free a slot and mark as not in-use (does not do any data copying)

---

Copyright 2017 The Apache Software Foundation.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_so

.

- 
- Extension
- so\_module
- mod\_so.c
- ( ) Base
- 

(DSO)

, ( .so ) , .so .d:

1.3 2.0 .

2.0



```
1.3.15 2.0 . mod_foo.so.
mod_so ApacheModuleFoo.dll ,
. 2.0 2.0 .
```

```
API . API
.
.
Configure ApacheCore ,
os\win32\modules.c .
```

```
LoadModule DLL
DLL
DLL . DLL module recor
() module record ( )
AP_MODULE_DECLARE_DATA . , :
```

```
module foo_module;
```

```
:
module AP_MODULE_DECLARE_DATA foo_module;
```

```
module record export .
DLL . libhttpd.dll libhttpd.lib e
. modules
. .dsp .dsp
.
```

DLL .

modules ,

LoadModu

.



## LoadFile

```
LoadFile filename [filename] ...  
Extension  
mod_so
```

LoadFile [\(link in\)](#).  
Filename [ServerRoot](#).

:

```
LoadFile libexec/libxmlparse.so
```



## LoadModule

```
LoadModule module filename
#
# Extension
# mod_so
```

LoadModule *filename* , *module*  
# *Module* *module* , *:*

```
LoadModule status_module modules/mod_status.so
```

ServerRoot modules .



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_socache\_dbm

<b>Description:</b>	DBM based shared object cache provider.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	socache_dbm_module
<b>Source File:</b>	mod_socache_dbm.c

## Summary

mod\_socache\_dbm is a shared object cache provider which provides for creation and access to a cache backed by a DBM database.

```
dbm:/path/to/datafile
```

Details of other shared object cache providers can be found [here](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_socache_dc`

<b>Description:</b>	Distcache based shared object cache provider.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>socache_dc_module</code>
<b>Source File:</b>	<code>mod_socache_dc.c</code>

## Summary

`mod_socache_dc` is a shared object cache provider which provides for creation and access to a cache backed by the [distcache](#) distributed session caching libraries.

Details of other shared object cache providers can be found [here](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_socache_memcache`

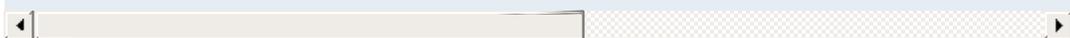
<b>Description:</b>	Memcache based shared object cache provider.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>socache_memcache_module</code>
<b>Source File:</b>	<code>mod_socache_memcache.c</code>

## Summary

`mod_socache_memcache` is a shared object cache provider which provides for creation and access to a cache backed by the [memcached](#) high-performance, distributed memory object caching system.

This shared object cache provider's "create" method requires a comma separated list of memcached host/port specifications. If using this provider via another modules configuration (such as [SSLSessionCache](#)), provide the list of servers as the optional "arg" parameter.

```
SSLSessionCache memcache:memcache.example.com:
```



Details of other shared object cache providers can be found [here](#).



<b>Description:</b>	Keepalive time for idle connections
<b>Syntax:</b>	MemcacheConnTTL <i>num[units]</i>
<b>Default:</b>	MemcacheConnTTL 15s
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_socache_memcache
<b>Compatibility:</b>	Available in Apache 2.4.17 and later

Set the time to keep idle connections with the memcache server(s) alive (threaded platforms only).

Valid values for **MemcacheConnTTL** are times up to one hour. 0 means no timeout.

This timeout defaults to units of seconds, but accepts suffixes for milliseconds (ms), seconds (s), minutes (min), and hours (h).

Before Apache 2.4.17, this timeout was hardcoded and its value was 600 usec. So, the closest configuration to match the legacy behaviour is to set **MemcacheConnTTL** to 1ms.

```
# Set a timeout of 10 minutes
MemcacheConnTTL 10min
# Set a timeout of 60 seconds
MemcacheConnTTL 60
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_socache_shmcb`

<b>Description:</b>	shmcb based shared object cache provider.
<b>Status:</b>	Extension
<b>Module Identifier:</b>	socache_shmcb_module
<b>Source File:</b>	mod_socache_shmcb.c

## Summary

`mod_socache_shmcb` is a shared object cache provider which provides for creation and access to a cache backed by a high-performance cyclic buffer inside a shared memory segment.

```
shmcb:/path/to/datafile(512000)
```

Details of other shared object cache providers can be found [here](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_speling

```
URL
Extension
speling_module
mod_speling.c
```

)

,

- , "document not found ( )" .
- "" ,
- ,



- Limits the action of the speling module to case corrections
- 
- , , directory, .htaccess
- Extension
- mod\_speling

Documentation not yet translated. Please see English version of document.



## CheckSpelling

```

:
:      CheckSpelling on|off
:      CheckSpelling Off
:      , , directory, .htaccess
Override : Options
:      Extension
:      mod_speling
:      1.1 CheckSpelling ,
:      . 1.3 . 1.3.2
:      CheckSpelling "" "" .

```

- 
- ""
- , (http://my.host/~
- . <Location /status>
- . "/stats.html"

[DAV](#) mod\_speling .  
doc34.html , DAV ""



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_ssl

<b>Description:</b>	Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols
<b>Status:</b>	Extension
<b>Module Identifier:</b>	ssl_module
<b>Source File:</b>	mod_ssl.c

## Summary

This module provides SSL v3 and TLS v1.x support for the Apache HTTP Server. SSL v2 is no longer supported.

This module relies on [OpenSSL](#) to provide the cryptography engine.

Further details, discussion, and examples are provided in the [SSL documentation](#).



This module can be configured to provide several items of SSL information as additional environment variables to the SSI and CGI namespace. This information is not provided by default for performance reasons. (See [SSLOptions StdEnvVars](#), below.) The generated variables are listed in the table below. For backward compatibility the information can be made available under different names, too. Look in the [Compatibility](#) chapter for details on the compatibility variables.

Variable Name:	Value Type:	Description:
HTTPS	flag	HTTPS is being used
SSL_PROTOCOL	string	The SSL protocol (SSLv3, TLSv1, TLSv1.1, TLSv1.2)
SSL_SESSION_ID	string	The hex-encoded session id
SSL_SESSION_RESUMED	string	Initial or Resumed Session. Note: multiple requests may be over the same (Initial or Resumed) SSL session. HTTP KeepAlive use
SSL_SECURE_RENEG	string	true if secure renegotiation is supported, else false
SSL_CIPHER	string	The cipher specification name
SSL_CIPHER_EXPORT	string	true if cipher is an export cipher

SSL_CIPHER_USEKEYSIZE	number	Number of cipher (actually used)
SSL_CIPHER_ALGKEYSIZE	number	Number of cipher (possible)
SSL_COMPRESS_METHOD	string	SSL compressor method negotiate
SSL_VERSION_INTERFACE	string	The mod_ssl pro version
SSL_VERSION_LIBRARY	string	The OpenSSL pro version
SSL_CLIENT_M_VERSION	string	The version of the certificate
SSL_CLIENT_M_SERIAL	string	The serial of the certificate
SSL_CLIENT_S_DN	string	Subject DN in client certificate
SSL_CLIENT_S_DN_x509	string	Component of client Subject DN
SSL_CLIENT_SAN_Email_n	string	Client certificate's subjectAltName extension entries rfc822Name
SSL_CLIENT_SAN_DNS_n	string	Client certificate's subjectAltName extension entries dNSName
SSL_CLIENT_SAN_OTHER_msUPN_n	string	Client certificate's subjectAltName extension entries otherName, Micro User Principal Name form (OID)

		1.3.6.1.4.1.311.20
SSL_CLIENT_I_DN	string	Issuer DN of client certificate
SSL_CLIENT_I_DN_x509	string	Component of client Issuer DN
SSL_CLIENT_V_START	string	Validity of client's certificate (start time)
SSL_CLIENT_V_END	string	Validity of client's certificate (end time)
SSL_CLIENT_V_REMAIN	string	Number of days until client's certificate expires
SSL_CLIENT_A_SIG	string	Algorithm used for signature of client certificate
SSL_CLIENT_A_KEY	string	Algorithm used for public key of client certificate
SSL_CLIENT_CERT	string	PEM-encoded client certificate
SSL_CLIENT_CERT_CHAIN_#	string	PEM-encoded certificates in client certificate chain
SSL_CLIENT_CERT_RFC4523_CEA	string	Serial number and format of the certificate. The format matches the CertificateExactA in RFC4523
SSL_CLIENT_VERIFY	string	NONE, SUCCESS, GENEROUS or FAILED: <i>reason</i>
SSL_SERVER_M_VERSION	string	The version of the

		certificate
SSL_SERVER_M_SERIAL	string	The serial of the : certificate
SSL_SERVER_S_DN	string	Subject DN in ser certificate
SSL_SERVER_SAN_Email_n	string	Server certificate' subjectAltName extension entries rfc822Name
SSL_SERVER_SAN_DNS_n	string	Server certificate' subjectAltName extension entries dNSName
SSL_SERVER_SAN_OTHER_dnsSRV_n	string	Server certificate' subjectAltName extension entries otherName, SRV form (OID 1.3.6.1.5.5.7.8.7, 4985)
SSL_SERVER_S_DN_x509	string	Component of se Subject DN
SSL_SERVER_I_DN	string	Issuer DN of serv certificate
SSL_SERVER_I_DN_x509	string	Component of se Issuer DN
SSL_SERVER_V_START	string	Validity of server' certificate (start ti
SSL_SERVER_V_END	string	Validity of server' certificate (end tir
SSL_SERVER_A_SIG	string	Algorithm used fo signature of serve

		certificate
SSL_SERVER_A_KEY	string	Algorithm used for public key of server certificate
SSL_SERVER_CERT	string	PEM-encoded server certificate
SSL_SRP_USER	string	SRP username
SSL_SRP_USERINFO	string	SRP user info
SSL_TLS_SNI	string	Contents of the SNI extension (if supported with ClientHello)

`x509` specifies a component of an X.509 DN; one of `C, ST, L, O, OU, CN, T, I, G, S, D, UID, Email`. In Apache 2.1 and later, `x509` may also include a numeric `_n` suffix. If the DN in question contains multiple attributes of the same name, this suffix is used as a zero-based index to select a particular attribute. For example, where the server certificate subject DN included two OU attributes, `SSL_SERVER_S_DN_OU_0` and `SSL_SERVER_S_DN_OU_1` could be used to reference each. A variable name without a `_n` suffix is equivalent to that name with a `_0` suffix; the first (or only) attribute. When the environment table is populated using the `StdEnvVars` option of the [SSLOptions](#) directive, the first (or only) attribute of any DN is added only under a non-suffixed name; i.e. no `_0` suffixed entries are added.

The format of the `*_DN` variables has changed in Apache HTTPD 2.3.11. See the `LegacyDNStringFormat` option for [SSLOptions](#) for details.

`SSL_CLIENT_V_REMAIN` is only available in version 2.1 and later.

A number of additional environment variables can also be used in

SSLRequire expressions, or in custom log formats:

HTTP_USER_AGENT	PATH_INFO	AUTH_TYPE
HTTP_REFERER	QUERY_STRING	SERVER_SOFTWARE
HTTP_COOKIE	REMOTE_HOST	API_VERSION
HTTP_FORWARDED	REMOTE_IDENT	TIME_YEAR
HTTP_HOST	IS_SUBREQ	TIME_MON
HTTP_PROXY_CONNECTION	DOCUMENT_ROOT	TIME_DAY
HTTP_ACCEPT	SERVER_ADMIN	TIME_HOUR
THE_REQUEST	SERVER_NAME	TIME_MIN
REQUEST_FILENAME	SERVER_PORT	TIME_SEC
REQUEST_METHOD	SERVER_PROTOCOL	TIME_WDAY
REQUEST_SCHEME	REMOTE_ADDR	TIME
REQUEST_URI	REMOTE_USER	

In these contexts, two special formats can also be used:

**ENV:*variablename***

This will expand to the standard environment variable *variablename*.

**HTTP:*headername***

This will expand to the value of the request header with name *headername*.



## Custom Log Format

When `mod_ssl` is built into Apache or at least loaded (under DSO situation) additional functions exist for the [Custom Log Format](#) of `mod_log_config`. First there is an additional ``%{varname}x"` extension format function which can be used to expand any variables provided by any module, especially those provided by `mod_ssl` which can you find in the above table.

For backward compatibility there is additionally a special ``%{name}c"` cryptography format function provided. Information about this function is provided in the [Compatibility](#) chapter.

### Example

```
CustomLog "logs/ssl_request_log" "%t %h %{SSL_PROTOCOL}x %{SSL_"
```

These formats even work without setting the `StdEnvVars` option of the [SSLOptions](#) directive.



## Request Notes

`mod_ssl` sets "notes" for the request which can be used in logging with the `%{name}n` format string in `mod_log_config`.

The notes supported are as follows:

### **ssl-access-forbidden**

This note is set to the value 1 if access was denied due to an `SSLRequire` or `SSLRequireSSL` directive.

### **ssl-secure-reneg**

If `mod_ssl` is built against a version of OpenSSL which supports the secure renegotiation extension, this note is set to the value 1 if SSL is in use for the current connection, and the client also supports the secure renegotiation extension. If the client does not support the secure renegotiation extension, the note is set to the value 0. If `mod_ssl` is not built against a version of OpenSSL which supports secure renegotiation, or if SSL is not in use for the current connection, the note is not set.



When `mod_ssl` is built into Apache or at least loaded (under DSO situation) any variables provided by `mod_ssl` can be used in expressions for the `ap_expr Expression Parser`. The variables can be referenced using the syntax ``#{varname}``. Starting with version 2.4.18 one can also use the `mod_rewrite` style syntax ``#{SSL:varname}`` or the function style syntax ``ssl(varname)``.

### Example (using `mod_headers`)

```
Header set X-SSL-PROTOCOL "expr=#{SSL_PROTOCOL}"  
Header set X-SSL-CIPHER "expr=#{SSL:SSL_CIPHER}"
```

This feature even works without setting the `StdEnvVars` option of the `SSLOptions` directive.



`mod_ssl` provides a few authentication providers for use with `mod_authz_core`'s `Require` directive.

## Require ssl

The `ssl` provider denies access if a connection is not encrypted with SSL. This is similar to the `SSLRequireSSL` directive.

```
Require ssl
```

## Require ssl-verify-client

The `ssl` provider allows access if the user is authenticated with a valid client certificate. This is only useful if `SSLVerifyClient optional` is in effect.

The following example grants access if the user is authenticated either with a client certificate or by username and password.

```
Require ssl-verify-client  
Require valid-user
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	SSLCACertificateFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLCACertificatePath](#).

### Example

```
SSLCACertificateFile "/usr/local/apache2/conf/ssl.crt/ca-bundle-
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	SSLCACertificatePath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose clients you deal with. These are used to verify the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value*.N. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCACertificatePath "/usr/local/apache2/conf/ssl.crt/"
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for defining acceptable CA names
<b>Syntax:</b>	SSLCADNRequestFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

When a client certificate is requested by mod\_ssl, a list of *acceptable Certificate Authority names* is sent to the client in the SSL handshake. These CA names can be used by the client to select an appropriate client certificate out of those it has available.

If neither of the directives [SSLCADNRequestPath](#) or [SSLCADNRequestFile](#) are given, then the set of acceptable CA names sent to the client is the names of all the CA certificates given by the [SSLCACertificateFile](#) and [SSLCACertificatePath](#) directives; in other words, the names of the CAs which will actually be used to verify the client certificate.

In some circumstances, it is useful to be able to send a set of acceptable CA names which differs from the actual CAs used to verify the client certificate - for example, if the client certificates are signed by intermediate CAs. In such cases, [SSLCADNRequestPath](#) and/or [SSLCADNRequestFile](#) can be used; the acceptable CA names are then taken from the complete set of certificates in the directory and/or file specified by this pair of directives.

[SSLCADNRequestFile](#) must specify an *all-in-one* file containing a concatenation of PEM-encoded CA certificates.

## Example

```
SSLCADNRequestFile "/usr/local/apache2/conf/ca-names.crt"
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for defining acceptable CA names
<b>Syntax:</b>	SSLCADNRequestPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This optional directive can be used to specify the set of *acceptable CA names* which will be sent to the client when a client certificate is requested. See the [SSLCADNRequestFile](#) directive for more details.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value.N*. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCADNRequestPath "/usr/local/apache2/conf/ca-names.crt/"
```



<b>Description:</b>	Enable CRL-based revocation checking
<b>Syntax:</b>	SSLCARevocationCheck chain leaf none <i>flags</i>
<b>Default:</b>	SSLCARevocationCheck none
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Optional <i>flags</i> available in httpd 2.4.21 or later

Enables certificate revocation list (CRL) checking. At least one of [SSLCARevocationFile](#) or [SSLCARevocationPath](#) must be configured. When set to chain (recommended setting), CRL checks are applied to all certificates in the chain, while setting it to leaf limits the checks to the end-entity cert.

The available *flags* are:

- `no_crl_for_cert_ok`  
Prior to version 2.3.15, CRL checking in mod\_ssl also succeeded when no CRL(s) for the checked certificate(s) were found in any of the locations configured with [SSLCARevocationFile](#) or [SSLCARevocationPath](#).

With the introduction of [SSLCARevocationFile](#), the behavior has been changed: by default with chain or leaf, CRLs **must** be present for the validation to succeed - otherwise it will fail with an "unable to get certificate CRL" error.

The *flag* `no_crl_for_cert_ok` allows to restore previous behaviour.

## Example

SSLCARevocationCheck chain

## Compatibility with versions 2.2

SSLCARevocationCheck chain no\_crl\_for\_cert\_ok



**Description:** File of concatenated PEM-encoded CA CRLs for Client Auth

**Syntax:** SSLCARevocationFile *file-path*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLCARevocationPath](#).

### Example

```
SSLCARevocationFile "/usr/local/apache2/conf/ssl.crl/ca-bundle-c
```



<b>Description:</b>	Directory of PEM-encoded CA CRLs for Client Auth
<b>Syntax:</b>	SSLCARevocationPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose clients you deal with. These are used to revoke the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCARevocationPath "/usr/local/apache2/conf/ssl.crl/"
```



<b>Description:</b>	File of PEM-encoded Server CA Certificates
<b>Syntax:</b>	<code>SSLCertificateChainFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ssl</code>

### SSLCertificateChainFile is deprecated

SSLCertificateChainFile became obsolete with version 2.4.8, when [SSLCertificateFile](#) was extended to also load intermediate CA certificates from the server certificate file.

This directive sets the optional *all-in-one* file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA Certificate files, usually in certificate chain order.

This should be used alternatively and/or additionally to [SSLCACertificatePath](#) for explicitly constructing the server certificate chain which is sent to the browser in addition to the server certificate. It is especially useful to avoid conflicts with CA certificates when using client authentication. Because although placing a CA certificate of the server certificate chain into [SSLCACertificatePath](#) has the same effect for the certificate chain construction, it has the side-effect that client certificates issued by this same CA certificate are also accepted on client authentication.

But be careful: Providing the certificate chain works only if you are using a *single* RSA or DSA based server certificate. If you are

using a coupled RSA+DSA certificate pair, this will work only if actually both certificates use the *same* certificate chain. Else the browsers will be confused in this situation.

### Example

```
SSLCertificateChainFile "/usr/local/apache2/conf/ssl.crt/ca.crt'
```



<b>Description:</b>	Server PEM-encoded X.509 certificate data file
<b>Syntax:</b>	<code>SSLCertificateFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ssl</code>

This directive points to a file with certificate data in PEM format. At a minimum, the file must include an end-entity (leaf) certificate. The directive can be used multiple times (referencing different filenames) to support multiple algorithms for server authentication - typically RSA, DSA, and ECC. The number of supported algorithms depends on the OpenSSL version being used for `mod_ssl`: with version 1.0.0 or later, `openssl list-public-key-algorithms` will output a list of supported algorithms, see also the note below about limitations of OpenSSL versions prior to 1.0.2 and the ways to work around them.

The files may also include intermediate CA certificates, sorted from leaf to root. This is supported with version 2.4.8 and later, and obsoletes [SSLCertificateChainFile](#). When running with OpenSSL 1.0.2 or later, this allows to configure the intermediate CA chain on a per-certificate basis.

Custom DH parameters and an EC curve name for ephemeral keys, can also be added to end of the first file configured using [SSLCertificateFile](#). This is supported in version 2.4.7 or later. Such parameters can be generated using the commands `openssl dhparam` and `openssl ecparam`. The parameters can be added as-is to the end of the first certificate file. Only the first file can be used for custom parameters, as they are applied independently of the authentication algorithm type.

Finally the end-entity certificate's private key can also be added to

the certificate file instead of using a separate [SSLCertificateKeyFile](#) directive. This practice is highly discouraged. If it is used, the certificate files using such an embedded key must be configured after the certificates using a separate key file. If the private key is encrypted, the pass phrase dialog is forced at startup time.

### **DH parameter interoperability with primes > 1024 bit**

Beginning with version 2.4.7, `mod_ssl` makes use of standardized DH parameters with prime lengths of 2048, 3072 and 4096 bits and with additional prime lengths of 6144 and 8192 bits beginning with version 2.4.10 (from [RFC 3526](#)), and hands them out to clients based on the length of the certificate's RSA/DSA key. With Java-based clients in particular (Java 7 or earlier), this may lead to handshake failures - see this [FAQ answer](#) for working around such issues.

### **Default DH parameters when using multiple certificates and OpenSSL versions prior to 1.0.2**

When using multiple certificates to support different authentication algorithms (like RSA, DSA, but mainly ECC) and OpenSSL prior to 1.0.2, it is recommended to either use custom DH parameters (preferably) by adding them to the first certificate file (as described above), or to order the [SSLCertificateFile](#) directives such that RSA/DSA certificates are placed **after** the ECC one.

This is due to a limitation in older versions of OpenSSL which don't let the Apache HTTP Server determine the currently selected certificate at handshake time (when the DH parameters must be sent to the peer) but instead always provide the last configured certificate. Consequently, the server may select default DH parameters based on the length of the wrong

certificate's key (ECC keys are much smaller than RSA/DSA ones and their length is not relevant for selecting DH primes).

Since custom DH parameters always take precedence over the default ones, this issue can be avoided by creating and configuring them (as described above), thus using a custom/suitable length.

### **Example**

```
SSLCertificateFile "/usr/local/apache2/conf/ssl.crt/server.crt"
```



<b>Description:</b>	Server PEM-encoded private key file
<b>Syntax:</b>	<code>SSLCertificateKeyFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

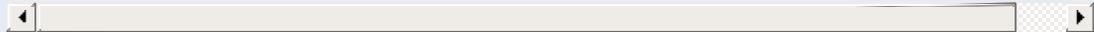
This directive points to the PEM-encoded private key file for the server. If the contained private key is encrypted, the pass phrase dialog is forced at startup time.

The directive can be used multiple times (referencing different filenames) to support multiple algorithms for server authentication. For each `SSLCertificateKeyFile` directive, there must be a matching `SSLCertificateFile` directive.

The private key may also be combined with the certificate in the file given by `SSLCertificateFile`, but this practice is highly discouraged. If it is used, the certificate files using such an embedded key must be configured after the certificates using a separate key file.

### Example

```
SSLCertificateKeyFile "/usr/local/apache2/conf/ssl.key/server.ke
```



<b>Description:</b>	Cipher Suite available for negotiation in SSL handshake
<b>Syntax:</b>	SSLCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLCipherSuite DEFAULT (depends on OpenSSL version)
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This complex directive uses a colon-separated *cipher-spec* string consisting of OpenSSL cipher specifications to configure the Cipher Suite the client is permitted to negotiate in the SSL handshake phase. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured Cipher Suite after the HTTP request was read but before the HTTP response is sent.

An SSL cipher specification in *cipher-spec* is composed of 4 major attributes plus a few extra minor ones:

- *Key Exchange Algorithm:*  
RSA, Diffie-Hellman, Elliptic Curve Diffie-Hellman, Secure Remote Password
- *Authentication Algorithm:*  
RSA, Diffie-Hellman, DSS, ECDSA, or none.
- *Cipher/Encryption Algorithm:*  
AES, DES, Triple-DES, RC4, RC2, IDEA, etc.
- *MAC Digest Algorithm:*  
MD5, SHA or SHA1, SHA256, SHA384.

An SSL cipher can also be an export cipher. SSLv2 ciphers are no longer supported. To specify which ciphers to use, one can either specify all the Ciphers, one at a time, or use aliases to specify the preference and order for the ciphers (see [Table 1](#)). The actually available ciphers and aliases depends on the used openssl version. Newer openssl versions may include additional ciphers.

<b>Tag</b>	<b>Description</b>
<i>Key Exchange Algorithm:</i>	
kRSA	RSA key exchange
kDHR	Diffie-Hellman key exchange with RSA key
kDHd	Diffie-Hellman key exchange with DSA key
kEDH	Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)
kSRP	Secure Remote Password (SRP) key exchange
<i>Authentication Algorithm:</i>	
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication
<i>Cipher Encoding Algorithm:</i>	
eNULL	No encryption
NULL	alias for eNULL
AES	AES encryption
DES	DES encryption
3DES	Triple-DES encryption
RC4	RC4 encryption
RC2	RC2 encryption
IDEA	IDEA encryption
<i>MAC Digest Algorithm:</i>	

MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	alias for SHA1
SHA256	SHA256 hash function
SHA384	SHA384 hash function
<i>Aliases:</i>	
SSLv3	all SSL version 3.0 ciphers
TLSv1	all TLS version 1.0 ciphers
EXP	all export ciphers
EXPORT40	all 40-bit export ciphers only
EXPORT56	all 56-bit export ciphers only
LOW	all low strength ciphers (no export, single DES)
MEDIUM	all ciphers with 128 bit encryption
HIGH	all ciphers using Triple-DES
RSA	all ciphers using RSA key exchange
DH	all ciphers using Diffie-Hellman key exchange
EDH	all ciphers using Ephemeral Diffie-Hellman key exchange
ECDH	Elliptic Curve Diffie-Hellman key exchange
ADH	all ciphers using Anonymous Diffie-Hellman key exchange
AECDH	all ciphers using Anonymous Elliptic Curve Diffie-Hellman key exchange
SRP	all ciphers using Secure Remote Password (SRP) key exchange
DSS	all ciphers using DSS authentication
ECDSA	all ciphers using ECDSA authentication
aNULL	all ciphers using no authentication

Now where this becomes interesting is that these can be put

together to specify the order and ciphers you wish to use. To speed this up there are also aliases (SSLv3, TLSv1, EXP, LOW, MEDIUM, HIGH) for certain groups of ciphers. These tags can be joined together with prefixes to form the *cipher-spec*. Available prefixes are:

- none: add cipher to list
- +: move matching ciphers to the current location in list
- -: remove cipher from list (can be added later again)
- !: kill cipher from list completely (can **not** be added later again)

### **aNULL, eNULL and EXP ciphers are always disabled**

Beginning with version 2.4.7, null and export-grade ciphers are always disabled, as `mod_ssl` unconditionally adds `!aNULL:!eNULL:!EXP` to any cipher string at initialization.

A simpler way to look at all of this is to use the `openssl ciphers -v` command which provides a nice way to successively create the correct *cipher-spec* string. The default *cipher-spec* string depends on the version of the OpenSSL libraries used. Let's suppose it is `RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5` which means the following: Put RC4-SHA and AES128-SHA at the beginning. We do this, because these ciphers offer a good compromise between speed and security. Next, include high and medium security ciphers. Finally, remove all ciphers which do not authenticate, i.e. for SSL the Anonymous Diffie-Hellman ciphers, as well as all ciphers which use MD5 as hash algorithm, because it has been proven insufficient.

```
$ openssl ciphers -v 'RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5'  
RC4-SHA          SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  M  
AES128-SHA       SSLv3 Kx=RSA      Au=RSA  Enc=AES(128)  M
```

DHE - RSA - AES256 - SHA	SSLv3 Kx=DH	Au=RSA	Enc=AES(256)	M
...	...	...	...	...
SEED - SHA	SSLv3 Kx=RSA	Au=RSA	Enc=SEED(128)	M
PSK - RC4 - SHA	SSLv3 Kx=PSK	Au=PSK	Enc=RC4(128)	M
KRB5 - RC4 - SHA	SSLv3 Kx=KRB5	Au=KRB5	Enc=RC4(128)	M

The complete list of particular RSA & DH ciphers for SSL is given in [Table 2](#).

### Example

SSLCipherSuite RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW

Cipher-Tag	Protocol	Key Ex.	Auth.	Enc.	MAC	Type
<i>RSA Ciphers:</i>						
DES - CBC3 - SHA	SSLv3	RSA	RSA	3DES(168)	SHA1	
IDEA - CBC - SHA	SSLv3	RSA	RSA	IDEA(128)	SHA1	
RC4 - SHA	SSLv3	RSA	RSA	RC4(128)	SHA1	
RC4 - MD5	SSLv3	RSA	RSA	RC4(128)	MD5	
DES - CBC - SHA	SSLv3	RSA	RSA	DES(56)	SHA1	
EXP - DES - CBC - SHA	SSLv3	RSA(512)	RSA	DES(40)	SHA1	export
EXP - RC2 - CBC - MD5	SSLv3	RSA(512)	RSA	RC2(40)	MD5	export
EXP - RC4 - MD5	SSLv3	RSA(512)	RSA	RC4(40)	MD5	export
NULL - SHA	SSLv3	RSA	RSA	None	SHA1	
NULL - MD5	SSLv3	RSA	RSA	None	MD5	
<i>Diffie-Hellman Ciphers:</i>						
ADH - DES -	SSLv3	DH	None	3DES(168)	SHA1	

CBC3 - SHA							
ADH - DES - CBC - SHA	SSLv3	DH	None	DES(56)	SHA1		
ADH - RC4 - MD5	SSLv3	DH	None	RC4(128)	MD5		
EDH - RSA - DES - CBC3 - SHA	SSLv3	DH	RSA	3DES(168)	SHA1		
EDH - DSS - DES - CBC3 - SHA	SSLv3	DH	DSS	3DES(168)	SHA1		
EDH - RSA - DES - CBC - SHA	SSLv3	DH	RSA	DES(56)	SHA1		
EDH - DSS - DES - CBC - SHA	SSLv3	DH	DSS	DES(56)	SHA1		
EXP - EDH - RSA - DES - CBC - SHA	SSLv3	DH(512)	RSA	DES(40)	SHA1	export	
EXP - EDH - DSS - DES - CBC - SHA	SSLv3	DH(512)	DSS	DES(40)	SHA1	export	
EXP - ADH - DES - CBC - SHA	SSLv3	DH(512)	None	DES(40)	SHA1	export	
EXP - ADH - RC4 - MD5	SSLv3	DH(512)	None	RC4(40)	MD5	export	



## SSLCompression Directive

<b>Description:</b>	Enable compression on the SSL level
<b>Syntax:</b>	SSLCompression on off
<b>Default:</b>	SSLCompression off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.3 and later, if using OpenSSL 0.9.8 or later; virtual host scope available if using OpenSSL 1.0.0 or later. The default used to be on in version 2.4.3.

This directive allows to enable compression on the SSL level.

Enabling compression causes security issues in most setups (the so called CRIME attack).



<b>Description:</b>	Enable use of a cryptographic hardware accelerator
<b>Syntax:</b>	SSLCryptoDevice <i>engine</i>
<b>Default:</b>	SSLCryptoDevice builtin
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive enables use of a cryptographic hardware accelerator board to offload some of the SSL processing overhead. This directive can only be used if the SSL toolkit is built with "engine" support; OpenSSL 0.9.7 and later releases have "engine" support by default, the separate "-engine" releases of OpenSSL 0.9.6 must be used.

To discover which engine names are supported, run the command "openssl engine".

### Example

```
# For a Broadcom accelerator:  
SSLCryptoDevice ubsec
```



<b>Description:</b>	SSL Engine Operation Switch
<b>Syntax:</b>	SSLEngine on off optional
<b>Default:</b>	SSLEngine off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine. This is should be used inside a [<VirtualHost>](#) section to enable SSL/TLS for a that virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts.

### Example

```
<VirtualHost _default_:443>  
SSLEngine on  
#...  
</VirtualHost>
```

In Apache 2.1 and later, [SSLEngine](#) can be set to `optional`. This enables support for [RFC 2817](#), Upgrading to TLS Within HTTP/1.1. At this time no web browsers support RFC 2817.



<b>Description:</b>	SSL FIPS mode Switch
<b>Syntax:</b>	SSLFIPS on off
<b>Default:</b>	SSLFIPS off
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL library FIPS\_mode flag. It must be set in the global server context and cannot be configured with conflicting settings (SSLFIPS on followed by SSLFIPS off or similar). The mode applies to all SSL library operations.

If httpd was compiled against an SSL library which did not support the FIPS\_mode flag, SSLFIPS on will fail. Refer to the FIPS 140-2 Security Policy document of the SSL provider library for specific requirements to use mod\_ssl in a FIPS 140-2 approved mode of operation; note that mod\_ssl itself is not validated, but may be described as using FIPS 140-2 validated cryptographic module, when all components are assembled and operated under the guidelines imposed by the applicable Security Policy.



## SSLHonorCipherOrder Directive

<b>Description:</b>	Option to prefer the server's cipher preference order
<b>Syntax:</b>	SSLHonorCipherOrder on off
<b>Default:</b>	SSLHonorCipherOrder off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

When choosing a cipher during an SSLv3 or TLSv1 handshake, normally the client's preference is used. If this directive is enabled, the server's preference will be used instead.

### Example

```
SSLHonorCipherOrder on
```



<b>Description:</b>	Option to enable support for insecure renegotiation
<b>Syntax:</b>	SSLInsecureRenegotiation on off
<b>Default:</b>	SSLInsecureRenegotiation off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.2.15 and later, if using OpenSSL 0.9.8m or later

As originally specified, all versions of the SSL and TLS protocols (up to and including TLS/1.2) were vulnerable to a Man-in-the-Middle attack ([CVE-2009-3555](#)) during a renegotiation. This vulnerability allowed an attacker to "prefix" a chosen plaintext to the HTTP request as seen by the web server. A protocol extension was developed which fixed this vulnerability if supported by both client and server.

If `mod_ssl` is linked against OpenSSL version 0.9.8m or later, by default renegotiation is only supported with clients supporting the new protocol extension. If this directive is enabled, renegotiation will be allowed with old (unpatched) clients, albeit insecurely.

### Security warning

If this directive is enabled, SSL connections will be vulnerable to the Man-in-the-Middle prefix attack as described in [CVE-2009-3555](#).

### Example

```
SSLInsecureRenegotiation on
```

The `SSL_SECURE_RENEG` environment variable can be used from an SSI or CGI script to determine whether secure renegotiation is supported for a given SSL connection.



<b>Description:</b>	Set the default responder URI for OCSP validation
<b>Syntax:</b>	<code>SSL0CSDefaultResponder <i>uri</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This option sets the default OCSP responder to use. If [SSL0CSPOverrideResponder](#) is not enabled, the URI given will be used only if no responder URI is specified in the certificate being verified.



**Description:** Enable OCSP validation of the client certificate chain

**Syntax:** SSLOCSPEnable on|off

**Default:** SSLOCSPEnable off

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_ssl

This option enables OCSP validation of the client certificate chain. If this option is enabled, certificates in the client's certificate chain will be validated against an OCSP responder after normal verification (including CRL checks) have taken place.

The OCSP responder used is either extracted from the certificate itself, or derived by configuration; see the [SSLOCSPDefaultResponder](#) and [SSLOCSPOverrideResponder](#) directives.

### Example

```
SSLVerifyClient on
SSLOCSPEnable on
SSLOCSPDefaultResponder "http://responder.example.com:8888/responder"
SSLOCSPOverrideResponder on
```



<b>Description:</b>	skip the OCSP responder certificates verification
<b>Syntax:</b>	SSL_OCSPVerify <i>On/Off</i>
<b>Default:</b>	SSL_OCSPVerify Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.26 and later, if using OpenSSL 0.9.7 or later

Skip the OCSP responder certificates verification, mostly useful when testing an OCSP server.



<b>Description:</b>	Force use of the default responder URI for OCSP validation
<b>Syntax:</b>	SSL <code>OCSP</code> overrideResponder on off
<b>Default:</b>	SSL <code>OCSP</code> overrideResponder off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This option forces the configured default OCSP responder to be used during OCSP certificate validation, regardless of whether the certificate being validated references an OCSP responder.



<b>Description:</b>	Proxy URL to use for OCSP requests
<b>Syntax:</b>	SSL0CSPProxyURL <i>url</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.19 and later

This option allows to set the URL of a HTTP proxy that should be used for all queries to OCSP responders.



<b>Description:</b>	Set of trusted PEM encoded OCSP responder certificates
<b>Syntax:</b>	SSL <code>OCSPResponderCertificateFile</code> <i>file</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.26 and later, if using OpenSSL 0.9.7 or later

This supplies a list of trusted OCSP responder certificates to be used during OCSP responder certificate validation. The supplied certificates are implicitly trusted without any further validation. This is typically used where the OCSP responder certificate is self signed or omitted from the OCSP response.



<b>Description:</b>	Timeout for OCSP queries
<b>Syntax:</b>	SSL0CSPResponderTimeout <i>seconds</i>
<b>Default:</b>	SSL0CSPResponderTimeout 10
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This option sets the timeout for queries to OCSP responders, when [SSL0CSPEnable](#) is turned on.



<b>Description:</b>	Maximum allowable age for OCSP responses
<b>Syntax:</b>	SSL0CSPPResponseMaxAge <i>seconds</i>
<b>Default:</b>	SSL0CSPPResponseMaxAge -1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This option sets the maximum allowable age ("freshness") for OCSP responses. The default value (-1) does not enforce a maximum age, which means that OCSP responses are considered valid as long as their `nextUpdate` field is in the future.



<b>Description:</b>	Maximum allowable time skew for OCSP response validation
<b>Syntax:</b>	SSL <code>OCSPResponseTimeSkew</code> <i>seconds</i>
<b>Default:</b>	SSL <code>OCSPResponseTimeSkew</code> 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This option sets the maximum allowable time skew for OCSP responses (when checking their `thisUpdate` and `nextUpdate` fields).



<b>Description:</b>	Use a nonce within OCSP queries
<b>Syntax:</b>	SSL0CSPUseRequestNonce on off
<b>Default:</b>	SSL0CSPUseRequestNonce on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.10 and later

This option determines whether queries to OCSP responders should contain a nonce or not. By default, a query nonce is always used and checked against the response's one. When the responder does not use nonces (e.g. Microsoft OCSP Responder), this option should be turned off.



<b>Description:</b>	Configure OpenSSL parameters through its <code>SSL_CONF</code> API
<b>Syntax:</b>	<code>SSLOpenSSLConfCmd</code> <i>command-name</i> <i>command-value</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ssl</code>
<b>Compatibility:</b>	Available in <code>httpd 2.4.8</code> and later, if using OpenSSL 1.0.2 or later

This directive exposes OpenSSL's `SSL_CONF` API to `mod_ssl`, allowing a flexible configuration of OpenSSL parameters without the need of implementing additional `mod_ssl` directives when new features are added to OpenSSL.

The set of available `SSLOpenSSLConfCmd` commands depends on the OpenSSL version being used for `mod_ssl` (at least version 1.0.2 is required). For a list of supported command names, see the section *Supported configuration file commands* in the [SSL\\_CONF\\_cmd\(3\)](#) manual page for OpenSSL.

Some of the `SSLOpenSSLConfCmd` commands can be used as an alternative to existing directives (such as `SSLCipherSuite` or `SSLProtocol`), though it should be noted that the syntax / allowable values for the parameters may sometimes differ.

## Examples

```
SSLOpenSSLConfCmd Options -SessionTicket,ServerPreference
SSLOpenSSLConfCmd ECDHParameters brainpoolP256r1
SSLOpenSSLConfCmd ServerInfoFile "/usr/local/apache2/conf/server
SSLOpenSSLConfCmd Protocol "-ALL, TLSv1.2"
SSLOpenSSLConfCmd SignatureAlgorithms RSA+SHA384:ECDSA+SHA256
```



<b>Description:</b>	Configure various SSL engine run-time options
<b>Syntax:</b>	SSLOptions [+ -] <i>option</i> ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control various run-time options on a per-directory basis. Normally, if multiple `SSLOptions` could apply to a directory, then the most specific one is taken completely; the options are not merged. However if *all* the options on the `SSLOptions` directive are preceded by a plus (+) or minus (-) symbol, the options are merged. Any options preceded by a + are added to the options currently in force, and any options preceded by a - are removed from the options currently in force.

The available *options* are:

- `StdEnvVars`  
When this option is enabled, the standard set of SSL related CGI/SSI environment variables are created. This per default is disabled for performance reasons, because the information extraction step is a rather expensive operation. So one usually enables this option for CGI and SSI requests only.
- `ExportCertData`  
When this option is enabled, additional CGI/SSI environment variables are created: `SSL_SERVER_CERT`, `SSL_CLIENT_CERT` and `SSL_CLIENT_CERT_CHAIN_n` (with  $n = 0,1,2,..$ ). These contain the PEM-encoded X.509 Certificates of server and client for the current HTTPS connection and can be used by CGI scripts for deeper Certificate checking. Additionally all other certificates of the

client certificate chain are provided, too. This bloats up the environment a little bit which is why you have to use this option to enable it on demand.

- **FakeBasicAuth**

When this option is enabled, the Subject Distinguished Name (DN) of the Client X509 Certificate is translated into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is just the Subject of the Client's X509 Certificate (can be determined by running OpenSSL's `openssl x509` command: `openssl x509 -noout -subject -in certificate.crt`). Note that no password is obtained from the user. Every entry in the user file needs this password: ``xxj31ZMTZzkVA"`, which is the DES-encrypted version of the word ``password"`. Those who live under MD5-based encryption (for instance under FreeBSD or BSD/OS, etc.) should use the following MD5 hash of the same word: ````$1$0XLYS...$0wx8s2/m9/gfkcRVXzgoE/"`.

Note that the [AuthBasicFake](#) directive within [mod\\_auth\\_basic](#) can be used as a more general mechanism for faking basic authentication, giving control over the structure of both the username and password.

- **StrictRequire**

This *forces* forbidden access when `SSLRequireSSL` or `SSLRequire` successfully decided that access should be forbidden. Usually the default is that in the case where a ````Satisfy any"` directive is used, and other access restrictions are passed, denial of access due to `SSLRequireSSL` or `SSLRequire` is overridden (because that's how the Apache `Satisfy` mechanism should work.) But for strict access restriction you can use `SSLRequireSSL`

and/or `SSLRequire` in combination with an `SSLOptions +StrictRequire`". Then an additional `Satisfy Any` has no chance once `mod_ssl` has decided to deny access.

- `OptRenegotiate`

This enables optimized SSL connection renegotiation handling when SSL directives are used in per-directory context. By default a strict scheme is enabled where *every* per-directory reconfiguration of SSL parameters causes a *full* SSL renegotiation handshake. When this option is used `mod_ssl` tries to avoid unnecessary handshakes by doing more granular (but still safe) parameter checks. Nevertheless these granular checks sometimes may not be what the user expects, so enable this on a per-directory basis only, please.

- `LegacyDNStringFormat`

This option influences how values of the `SSL_{CLIENT, SERVER}_{I, S}_DN` variables are formatted. Since version 2.3.11, Apache HTTPD uses a RFC 2253 compatible format by default. This uses commas as delimiters between the attributes, allows the use of non-ASCII characters (which are converted to UTF8), escapes various special characters with backslashes, and sorts the attributes with the "C" attribute last.

If `LegacyDNStringFormat` is set, the old format will be used which sorts the "C" attribute first, uses slashes as separators, and does not handle non-ASCII and special characters in any consistent way.

### Example

```
SSLOptions +FakeBasicAuth -StrictRequire
<Files ~ "\.(cgi|shtml)$">
    SSLOptions +StdEnvVars -ExportCertData
</Files>
```



<b>Description:</b>	Type of pass phrase dialog for encrypted private keys
<b>Syntax:</b>	SSLPassPhraseDialog <i>type</i>
<b>Default:</b>	SSLPassPhraseDialog builtin
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

When Apache starts up it has to read the various Certificate (see [SSLCertificateFile](#)) and Private Key (see [SSLCertificateKeyFile](#)) files of the SSL-enabled virtual servers. Because for security reasons the Private Key files are usually encrypted, mod\_ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. This query can be done in two ways which can be configured by *type*:

- **builtin**  
This is the default where an interactive terminal dialog occurs at startup time just before Apache detaches from the terminal. Here the administrator has to manually enter the Pass Phrase for each encrypted Private Key file. Because a lot of SSL-enabled virtual hosts can be configured, the following reuse-scheme is used to minimize the dialog: When a Private Key file is encrypted, all known Pass Phrases (at the beginning there are none, of course) are tried. If one of those known Pass Phrases succeeds no dialog pops up for this particular Private Key file. If none succeeded, another Pass Phrase is queried on the terminal and remembered for the next round (where it perhaps can be reused).

This scheme allows mod\_ssl to be maximally flexible (because for N encrypted Private Key files you *can* use N different Pass Phrases - but then you have to enter all of

them, of course) while minimizing the terminal dialog (i.e. when you use a single Pass Phrase for all N Private Key files this Pass Phrase is queried only once).

- `|/path/to/program [args...]`  
This mode allows an external program to be used which acts as a pipe to a particular input device; the program is sent the standard prompt text used for the `builtin` mode on `stdin`, and is expected to write password strings on `stdout`. If several passwords are needed (or an incorrect password is entered), additional prompt text will be written subsequent to the first password being returned, and more passwords must then be written back.
- `exec:/path/to/program`  
Here an external program is configured which is called at startup for each encrypted Private Key file. It is called with two arguments (the first is of the form `servername:portnumber`, the second is either `RSA`, `DSA`, `ECC` or an integer index starting at 3 if more than three keys are configured), which indicate for which server and algorithm it has to print the corresponding Pass Phrase to `stdout`. In versions 2.4.8 (unreleased) and 2.4.9, it is called with one argument, a string of the form `servername:portnumber:index` (with `index` being a zero-based integer number), which indicate the server, TCP port and certificate number. The intent is that this external program first runs security checks to make sure that the system is not compromised by an attacker, and only when these checks were passed successfully it provides the Pass Phrase.

Both these security checks, and the way the Pass Phrase is determined, can be as complex as you like. `Mod_ssl` just

defines the interface: an executable program which provides the Pass Phrase on stdout. Nothing more or less! So, if you're really paranoid about security, here is your interface. Anything else has to be left as an exercise to the administrator, because local security requirements are so different.

The reuse-algorithm above is used here, too. In other words: The external program is called only once per unique Pass Phrase.

### **Example**

```
SSLPassPhraseDialog "exec:/usr/local/apache/sbin/pp-filter"
```



<b>Description:</b>	Configure usable SSL/TLS protocol versions
<b>Syntax:</b>	SSLProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProtocol all -SSLv3 (up to 2.4.16: all)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control which versions of the SSL/TLS protocol will be accepted in new connections.

The available (case-insensitive) *protocols* are:

- SSLv3  
This is the Secure Sockets Layer (SSL) protocol, version 3.0, from the Netscape Corporation. It is the successor to SSLv2 and the predecessor to TLSv1, but is deprecated in [RFC 7568](#).
- TLSv1  
This is the Transport Layer Security (TLS) protocol, version 1.0. It is the successor to SSLv3 and is defined in [RFC 2246](#). It is supported by nearly every client.
- TLSv1.1 (when using OpenSSL 1.0.1 and later)  
A revision of the TLS 1.0 protocol, as defined in [RFC 4346](#).
- TLSv1.2 (when using OpenSSL 1.0.1 and later)  
A revision of the TLS 1.1 protocol, as defined in [RFC 5246](#).
- all  
This is a shortcut for ``+SSLv3 +TLSv1" or - when using OpenSSL 1.0.1 and later - ``+SSLv3 +TLSv1 +TLSv1.1

+TLSv1.2", respectively (except for OpenSSL versions compiled with the ``no-ssl3" configuration option, where all does not include +SSLv3).

### Example

```
SSLProtocol TLSv1
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Remote Server Auth
<b>Syntax:</b>	SSLProxyCACertificateFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCACertificatePath](#).

### Example

```
SSLProxyCACertificateFile "/usr/local/apache2/conf/ssl.crt/ca-bi
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for Remote Server Auth
<b>Syntax:</b>	SSLProxyCACertificatePath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose remote servers you deal with. These are used to verify the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value*.N. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLProxyCACertificatePath "/usr/local/apache2/conf/ssl.crt/"
```



<b>Description:</b>	Enable CRL-based revocation checking for Remote Server Auth
<b>Syntax:</b>	SSLProxyCAREvocationCheck chain leaf none
<b>Default:</b>	SSLProxyCAREvocationCheck none
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

Enables certificate revocation list (CRL) checking for the *remote servers* you deal with. At least one of [SSLProxyCAREvocationFile](#) or [SSLProxyCAREvocationPath](#) must be configured. When set to `chain` (recommended setting), CRL checks are applied to all certificates in the chain, while setting it to `leaf` limits the checks to the end-entity cert.

### **When set to `chain` or `leaf`, CRLs *must* be available for successful validation**

Prior to version 2.3.15, CRL checking in `mod_ssl` also succeeded when no CRL(s) were found in any of the locations configured with [SSLProxyCAREvocationFile](#) or [SSLProxyCAREvocationPath](#). With the introduction of this directive, the behavior has been changed: when checking is enabled, CRLs *must* be present for the validation to succeed - otherwise it will fail with an "unable to get certificate CRL" error.

### **Example**

```
SSLProxyCAREvocationCheck chain
```



**Description:** File of concatenated PEM-encoded CA CRLs for Remote Server Auth

**Syntax:** SSLProxyCARevocationFile *file-path*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCARevocationPath](#).

### Example

```
SSLProxyCARevocationFile "/usr/local/apache2/conf/ssl.crl/ca-bur
```



<b>Description:</b>	Directory of PEM-encoded CA CRLs for Remote Server Auth
<b>Syntax:</b>	SSLProxyCAREvocationPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose remote servers you deal with. These are used to revoke the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLProxyCAREvocationPath "/usr/local/apache2/conf/ssl.crl/"
```



<b>Description:</b>	Whether to check the remote server certificate's CN field
<b>Syntax:</b>	SSLProxyCheckPeerCN on off
<b>Default:</b>	SSLProxyCheckPeerCN on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets whether the remote server certificate's CN field is compared against the hostname of the request URL. If both are not equal a 502 status code (Bad Gateway) is sent.

SSLProxyCheckPeerCN is superseded by [SSLProxyCheckPeerName](#) in release 2.4.5 and later.

In all releases 2.4.5 through 2.4.20, setting `SSLProxyCheckPeerName off` was sufficient to enable this behavior (as the `SSLProxyCheckPeerCN` default was on.) In these releases, both directives must be set to `off` to completely avoid remote server certificate name validation. Many users reported this to be very confusing.

As of release 2.4.21, all configurations which enable either one of the `SSLProxyCheckPeerName` or `SSLProxyCheckPeerCN` options will use the new [SSLProxyCheckPeerName](#) behavior, and all configurations which disable either one of the `SSLProxyCheckPeerName` or `SSLProxyCheckPeerCN` options will suppress all remote server certificate name validation. Only the following configuration will trigger the legacy certificate CN comparison in 2.4.21 and later releases;

### Example

```
SSLProxyCheckPeerCN on
```

SSLProxyCheckPeerName off



## SSLProxyCheckPeerExpire Directive

<b>Description:</b>	Whether to check if remote server certificate is expired
<b>Syntax:</b>	SSLProxyCheckPeerExpire on off
<b>Default:</b>	SSLProxyCheckPeerExpire on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets whether it is checked if the remote server certificate is expired or not. If the check fails a 502 status code (Bad Gateway) is sent.

**Example**

```
SSLProxyCheckPeerExpire on
```



<b>Description:</b>	Configure host name checking for remote server certificates
<b>Syntax:</b>	SSLProxyCheckPeerName on off
<b>Default:</b>	SSLProxyCheckPeerName on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Apache HTTP Server 2.4.5 and later

This directive configures host name checking for server certificates when mod\_ssl is acting as an SSL client. The check will succeed if the host name from the request URI matches one of the CN attribute(s) of the certificate's subject, or matches the subjectAltName extension. If the check fails, the SSL request is aborted and a 502 status code (Bad Gateway) is returned.

Wildcard matching is supported for specific cases: an subjectAltName entry of type dNSName, or CN attributes starting with \* . will match with any host name of the same number of name elements and the same suffix. E.g. \* .example .org will match foo .example .org, but will not match foo .bar .example .org, because the number of elements in the respective host names differs.

This feature was introduced in 2.4.5 and superseded the behavior of the [SSLProxyCheckPeerCN](#) directive, which only tested the exact value in the first CN attribute against the host name. However, many users were confused by the behavior of using these directives individually, so the mutual behavior of SSLProxyCheckPeerName and SSLProxyCheckPeerCN directives were improved in release 2.4.21. See the [SSLProxyCheckPeerCN](#) directive description for the original

behavior and details of these improvements.



## SSLProxyCipherSuite Directive

<b>Description:</b>	Cipher Suite available for negotiation in SSL proxy handshake
<b>Syntax:</b>	SSLProxyCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+E
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

Equivalent to [SSLCipherSuite](#), but for the proxy connection. Please refer to [SSLCipherSuite](#) for additional information.



## SSLProxyEngine Directive

<b>Description:</b>	SSL Proxy Engine Operation Switch
<b>Syntax:</b>	SSLProxyEngine on off
<b>Default:</b>	SSLProxyEngine off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine for proxy. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for proxy usage in a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for proxy both for the main server and all configured virtual hosts.

Note that the `SSLProxyEngine` directive should not, in general, be included in a virtual host that will be acting as a forward proxy (using `<Proxy>` or `ProxyRequests` directives). `SSLProxyEngine` is not required to enable a forward proxy server to proxy SSL/TLS requests.

### Example

```
<VirtualHost _default_:443>
  SSLProxyEngine on
  #...
</VirtualHost>
```



<b>Description:</b>	File of concatenated PEM-encoded CA certificates to be used by the proxy for choosing a certificate
<b>Syntax:</b>	SSLProxyMachineCertificateChainFile <i>filename</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the all-in-one file where you keep the certificate chain for all of the client certs in use. This directive will be needed if the remote server presents a list of CA certificates that are not direct signers of one of the configured client certificates.

This referenced file is simply the concatenation of the various PEM-encoded certificate files. Upon startup, each client certificate configured will be examined and a chain of trust will be constructed.

### Security warning

If this directive is enabled, all of the certificates in the file will be trusted as if they were also in [SSLProxyCACertificateFile](#).

### Example

```
SSLProxyMachineCertificateChainFile "/usr/local/apache2/conf/ss:  
◀ [ ] ▶
```



<b>Description:</b>	File of concatenated PEM-encoded client certificates and keys to be used by the proxy
<b>Syntax:</b>	SSLProxyMachineCertificateFile <i>filename</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the all-in-one file where you keep the certificates and keys used for authentication of the proxy server to remote servers.

This referenced file is simply the concatenation of the various PEM-encoded certificate files, in order of preference. Use this directive alternatively or additionally to SSLProxyMachineCertificatePath.

Currently there is no support for encrypted private keys

### Example

```
SSLProxyMachineCertificateFile "/usr/local/apache2/conf/ssl.crt,
```



<b>Description:</b>	Directory of PEM-encoded client certificates and keys to be used by the proxy
<b>Syntax:</b>	SSLProxyMachineCertificatePath <i>directory</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the certificates and keys used for authentication of the proxy server to remote servers.

The files in this directory must be PEM-encoded and are accessed through hash filenames. Additionally, you must create symbolic links named *hash-value.N*. And you should always make sure this directory contains the appropriate symbolic links.

Currently there is no support for encrypted private keys

### Example

```
SSLProxyMachineCertificatePath "/usr/local/apache2/conf/proxy.c
```



<b>Description:</b>	Configure usable SSL protocol flavors for proxy usage
<b>Syntax:</b>	SSLProxyProtocol [ <b>+</b>   <b>-</b> ]protocol ...
<b>Default:</b>	SSLProxyProtocol all -SSLv3 (up to 2.4.16: all)
<b>Context:</b>	server config, virtual host
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control the SSL protocol flavors mod\_ssl should use when establishing its server environment for proxy . It will only connect to servers using one of the provided protocols.

Please refer to [SSLProtocol](#) for additional information.



## SSLProxyVerify Directive

<b>Description:</b>	Type of remote server Certificate verification
<b>Syntax:</b>	SSLProxyVerify <i>level</i>
<b>Default:</b>	SSLProxyVerify none
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

When a proxy is configured to forward requests to a remote SSL server, this directive can be used to configure certificate verification of the remote server.

The following levels are available for *level*:

- **none**: no remote server Certificate is required at all
- **optional**: the remote server *may* present a valid Certificate
- **require**: the remote server *has to* present a valid Certificate
- **optional\_no\_ca**: the remote server may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all servers and level **optional\_no\_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

### Example

```
SSLProxyVerify require
```



## SSLProxyVerifyDepth Directive

<b>Description:</b>	Maximum depth of CA Certificates in Remote Server Certificate verification
<b>Syntax:</b>	SSLProxyVerifyDepth <i>number</i>
<b>Default:</b>	SSLProxyVerifyDepth 1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets how deeply mod\_ssl should verify before deciding that the remote server does not have a valid certificate.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the remote server certificate. A depth of 0 means that self-signed remote server certificates are accepted only, the default depth of 1 means the remote server certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under [SSLProxyCACertificatePath](#)), etc.

**Example**

```
SSLProxyVerifyDepth 10
```



<b>Description:</b>	Pseudo Random Number Generator (PRNG) seeding source
<b>Syntax:</b>	SSLRandomSeed <i>context source [bytes]</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures one or more sources for seeding the Pseudo Random Number Generator (PRNG) in OpenSSL at startup time (*context* is `start up`) and/or just before a new SSL connection is established (*context* is `connect`). This directive can only be used in the global server context because the PRNG is a global facility.

The following *source* variants are available:

- `builtin`  
This is the always available builtin seeding source. Its usage consumes minimum CPU cycles under runtime and hence can be always used without drawbacks. The source used for seeding the PRNG contains of the current time, the current process id and (when applicable) a randomly chosen 1KB extract of the inter-process scoreboard structure of Apache. The drawback is that this is not really a strong source and at startup time (where the scoreboard is still not available) this source just produces a few bytes of entropy. So you should always, at least for the startup, use an additional seeding source.
- `file:/path/to/source`  
This variant uses an external file `/path/to/source` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of the file form the entropy (and *bytes* is given to `/path/to/source` as the first

argument). When *bytes* is not specified the whole file forms the entropy (and 0 is given to `/path/to/source` as the first argument). Use this especially at startup time, for instance with an available `/dev/random` and/or `/dev/urandom` devices (which usually exist on modern Unix derivatives like FreeBSD and Linux).

*But be careful:* Usually `/dev/random` provides only as much entropy data as it actually has, i.e. when you request 512 bytes of entropy, but the device currently has only 100 bytes available two things can happen: On some platforms you receive only the 100 bytes while on other platforms the read blocks until enough bytes are available (which can take a long time). Here using an existing `/dev/urandom` is better, because it never blocks and actually gives the amount of requested data. The drawback is just that the quality of the received data may not be the best.

- `exec:/path/to/program`  
This variant uses an external executable `/path/to/program` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of its `stdout` contents form the entropy. When *bytes* is not specified, the entirety of the data produced on `stdout` form the entropy. Use this only at startup time when you need a very strong seeding with the help of an external program (for instance as in the example above with the `truerand` utility you can find in the `mod_ssl` distribution which is based on the AT&T *truerand* library). Using this in the connection context slows down the server too dramatically, of course. So usually you should avoid using external programs in that context.
- `egd:/path/to/egd-socket` (Unix only)  
This variant uses the Unix domain socket of the external

Entropy Gathering Daemon (EGD) (see <http://www.lothar.com/tech/crypto/>) to seed the PRNG. Use this if no random device exists on your platform.

### Example

```
SSLRandomSeed startup builtin
SSLRandomSeed startup "file:/dev/random"
SSLRandomSeed startup "file:/dev/urandom" 1024
SSLRandomSeed startup "exec:/usr/local/bin/truerand" 16
SSLRandomSeed connect builtin
SSLRandomSeed connect "file:/dev/random"
SSLRandomSeed connect "file:/dev/urandom" 1024
```



<b>Description:</b>	Set the size for the SSL renegotiation buffer
<b>Syntax:</b>	SSLRenegBufferSize <i>bytes</i>
<b>Default:</b>	SSLRenegBufferSize 131072
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

If an SSL renegotiation is required in per-location context, for example, any use of [SSLVerifyClient](#) in a Directory or Location block, then [mod\\_ssl](#) must buffer any HTTP request body into memory until the new SSL handshake can be performed. This directive can be used to set the amount of memory that will be used for this buffer.

Note that in many configurations, the client sending the request body will be untrusted so a denial of service attack by consumption of memory must be considered when changing this configuration setting.

### Example

```
SSLRenegBufferSize 262144
```



<b>Description:</b>	Allow access only when an arbitrarily complex boolean expression is true
<b>Syntax:</b>	SSLRequire <i>expression</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

### SSLRequire is deprecated

SSLRequire is deprecated and should in general be replaced by [Require expr](#). The so called [ap\\_expr](#) syntax of Require expr is a superset of the syntax of SSLRequire, with the following exception:

In SSLRequire, the comparison operators <, <=, ... are completely equivalent to the operators lt, le, ... and work in a somewhat peculiar way that first compares the length of two strings and then the lexical order. On the other hand, [ap\\_expr](#) has two sets of comparison operators: The operators <, <=, ... do lexical string comparison, while the operators -lt, -le, ... do integer comparison. For the latter, there are also aliases without the leading dashes: lt, le, ...

This directive specifies a general access requirement which has to be fulfilled in order to allow access. It is a very powerful directive because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks.

The *expression* must match the following syntax (given as a BNF grammar notation):

```
expr      ::= "true" | "false"
```

```

| "!" expr
| expr "&&" expr
| expr "||" expr
| "(" expr ")"
| comp

comp ::= word "==" word | word "eq" word
      | word "!=" word | word "ne" word
      | word "<" word | word "lt" word
      | word "<=" word | word "le" word
      | word ">" word | word "gt" word
      | word ">=" word | word "ge" word
      | word "in" "{" wordlist "}"
      | word "in" "PeerExtList(" word ")"
      | word "=~" regex
      | word "!~" regex

wordlist ::= word
          | wordlist "," word

word ::= digit
      | cstring
      | variable
      | function

digit ::= [0-9]+
cstring ::= "... "
variable ::= "%{" varname "}"
function ::= funcname "(" funcargs ")"

```

For varname any of the variables described in [Environment Variables](#) can be used. For funcname the available functions are listed in the [ap\\_expr documentation](#).

The *expression* is parsed into an internal machine representation when the configuration is loaded, and then evaluated during request processing. In .htaccess context, the *expression* is both

parsed and executed each time the .htaccess file is encountered during request processing.

### Example

```
SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)-/  
              and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd."  
              and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"}  
              and %{TIME_WDAY} -ge 1 and %{TIME_WDAY} -le 5  
              and %{TIME_HOUR} -ge 8 and %{TIME_HOUR} -le 20  
              or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
```

The `PeerExtList(object-ID)` function expects to find zero or more instances of the X.509 certificate extension identified by the given *object ID* (OID) in the client certificate. The expression evaluates to true if the left-hand side string matches exactly against the value of an extension identified with this OID. (If multiple extensions with the same OID are present, at least one extension must match).

### Example

```
SSLRequire "foobar" in PeerExtList("1.2.3.4.5.6")
```

### Notes on the PeerExtList function

- The object ID can be specified either as a descriptive name recognized by the SSL library, such as "nsComment", or as a numeric OID, such as "1.2.3.4.5.6".
- Expressions with types known to the SSL library are rendered to a string before comparison. For an extension with a type not recognized by the SSL library, `mod_ssl` will parse the value if it is one of the primitive ASN.1 types UTF8String, IA5String, VisibleString, or BMPString. For an extension of one of these types, the string value will be converted to UTF-8 if necessary, then compared against

the left-hand-side expression.

## See also

- [Environment Variables in Apache HTTP Server](#), for additional examples.
- [Require expr](#)
- [Generic expression syntax in Apache HTTP Server](#)



<b>Description:</b>	Deny access when SSL is not used for the HTTP request
<b>Syntax:</b>	SSLRequireSSL
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive forbids access unless HTTP over SSL (i.e. HTTPS) is enabled for the current connection. This is very handy inside the SSL-enabled virtual host or directories for defending against configuration errors that expose stuff that should be protected. When this directive is present all requests are denied which are not using SSL.

### Example

```
SSLRequireSSL
```



<b>Description:</b>	Type of the global/inter-process SSL Session Cache
<b>Syntax:</b>	SSLSessionCache <i>type</i>
<b>Default:</b>	SSLSessionCache none
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures the storage type of the global/inter-process SSL Session Cache. This cache is an optional facility which speeds up parallel request processing. For requests to the same server process (via HTTP keep-alive), OpenSSL already caches the SSL session information locally. But because modern clients request inlined images and other data via parallel requests (usually up to four parallel requests are common) those requests are served by *different* pre-forked server processes. Here an inter-process cache helps to avoid unnecessary session handshakes.

The following five storage *types* are currently supported:

- none  
This disables the global/inter-process Session Cache. This will incur a noticeable speed penalty and may cause problems if using certain browsers, particularly if client certificates are enabled. This setting is not recommended.
- nonenotnull  
This disables any global/inter-process Session Cache. However it does force OpenSSL to send a non-null session ID to accommodate buggy clients that require one.
- dbm:/path/to/datafile  
This makes use of a DBM hashfile on the local disk to

synchronize the local OpenSSL memory caches of the server processes. This session cache may suffer reliability issues under high load. To use this, ensure that [mod\\_socache\\_dbm](#) is loaded.

- `shmcb:/path/to/datafile[(size)]`  
This makes use of a high-performance cyclic buffer (approx. *size* bytes in size) inside a shared memory segment in RAM (established via `/path/to/datafile`) to synchronize the local OpenSSL memory caches of the server processes. This is the recommended session cache. To use this, ensure that [mod\\_socache\\_shmcb](#) is loaded.
- `dc:UNIX:/path/to/socket`  
This makes use of the [distcache](#) distributed session caching libraries. The argument should specify the location of the server or proxy to be used using the distcache address syntax; for example, `UNIX:/path/to/socket` specifies a UNIX domain socket (typically a local `dc_client` proxy); `IP:server.example.com:9001` specifies an IP address. To use this, ensure that [mod\\_socache\\_dc](#) is loaded.

## Examples

```
SSLSessionCache "dbm:/usr/local/apache/logs/ssl_gcach_data"  
SSLSessionCache "shmcb:/usr/local/apache/logs/ssl_gcach_data(5:  
◀────────────────────────────────────────────────────────────────────────────────▶
```

The `ssl-cache` mutex is used to serialize access to the session cache to prevent corruption. This mutex can be configured using the [Mutex](#) directive.



<b>Description:</b>	Number of seconds before an SSL session expires in the Session Cache
<b>Syntax:</b>	SSLSessionCacheTimeout <i>seconds</i>
<b>Default:</b>	SSLSessionCacheTimeout 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Applies also to RFC 5077 TLS session resumption in Apache 2.4.10 and later

This directive sets the timeout in seconds for the information stored in the global/inter-process SSL Session Cache, the OpenSSL internal memory cache and for sessions resumed by TLS session resumption (RFC 5077). It can be set as low as 15 for testing, but should be set to higher values like 300 in real life.

### Example

```
SSLSessionCacheTimeout 600
```



<b>Description:</b>	Persistent encryption/decryption key for TLS session tickets
<b>Syntax:</b>	SSLSessionTicketKeyFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.0 and later, if using OpenSSL 0.9.8h or later

Optionally configures a secret key for encrypting and decrypting TLS session tickets, as defined in [RFC 5077](#). Primarily suitable for clustered environments where TLS sessions information should be shared between multiple nodes. For single-instance httpd setups, it is recommended to *not* configure a ticket key file, but to rely on (random) keys generated by mod\_ssl at startup, instead.

The ticket key file must contain 48 bytes of random data, preferably created from a high-entropy source. On a Unix-based system, a ticket key file can be created as follows:

```
dd if=/dev/random of=/path/to/file.tkey bs=1 count=48
```

Ticket keys should be rotated (replaced) on a frequent basis, as this is the only way to invalidate an existing session ticket - OpenSSL currently doesn't allow to specify a limit for ticket lifetimes. A new ticket key only gets used after restarting the web server. All existing session tickets become invalid after a restart.

The ticket key file contains sensitive keying material and should be protected with file permissions similar to those used for [SSLCertificateKeyFile](#).



<b>Description:</b>	Enable or disable use of TLS session tickets
<b>Syntax:</b>	SSLSessionTickets on off
<b>Default:</b>	SSLSessionTickets on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.11 and later, if using OpenSSL 0.9.8f or later.

This directive allows to enable or disable the use of TLS session tickets (RFC 5077).

TLS session tickets are enabled by default. Using them without restarting the web server with an appropriate frequency (e.g. daily) compromises perfect forward secrecy.



<b>Description:</b>	SRP unknown user seed
<b>Syntax:</b>	SSLSRPUnknownUserSeed <i>secret-string</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.4 and later, if using OpenSSL 1.0.1 or later

This directive sets the seed used to fake SRP user parameters for unknown users, to avoid leaking whether a given user exists. Specify a secret string. If this directive is not used, then Apache will return the UNKNOWN\_PSK\_IDENTITY alert to clients who specify an unknown username.

### Example

```
SSLSRPUnknownUserSeed "secret"
```



<b>Description:</b>	Path to SRP verifier file
<b>Syntax:</b>	SSLSRPVerifierFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.4.4 and later, if using OpenSSL 1.0.1 or later

This directive enables TLS-SRP and sets the path to the OpenSSL SRP (Secure Remote Password) verifier file containing TLS-SRP usernames, verifiers, salts, and group parameters.

### Example

```
SSLSRPVerifierFile "/path/to/file.srpv"
```

The verifier file can be created with the `openssl` command line utility:

### Creating the SRP verifier file

```
openssl srp -srpvfile passwd.srpv -userinfo "some info" -add  
username
```

The value given with the optional `-userinfo` parameter is available in the `SSL_SRP_USERINFO` request environment variable.



<b>Description:</b>	Configures the OCSP stapling cache
<b>Syntax:</b>	SSLStaplingCache <i>type</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

Configures the cache used to store OCSP responses which get included in the TLS handshake if [SSLUseStapling](#) is enabled. Configuration of a cache is mandatory for OCSP stapling. With the exception of none and nonenotnull, the same storage types are supported as with [SSLSessionCache](#).



## SSLStaplingErrorCacheTimeout

<b>Description:</b>	Number of seconds before expiring invalid responses in the OCSP stapling cache
<b>Syntax:</b>	SSLStaplingErrorCacheTimeout <i>seconds</i>
<b>Default:</b>	SSLStaplingErrorCacheTimeout 600
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

Sets the timeout in seconds before *invalid* responses in the OCSP stapling cache (configured through [SSLStaplingCache](#)) will expire. To set the cache timeout for valid responses, see [SSLStaplingStandardCacheTimeout](#).



## SSLStaplingFakeTryLater Directive

<b>Description:</b>	Synthesize "tryLater" responses for failed OCSP stapling queries
<b>Syntax:</b>	SSLStaplingFakeTryLater on off
<b>Default:</b>	SSLStaplingFakeTryLater on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

When enabled and a query to an OCSP responder for stapling purposes fails, mod\_ssl will synthesize a "tryLater" response for the client. Only effective if [SSLStaplingReturnResponderErrors](#) is also enabled.



## SSLStaplingForceURL Directive

<b>Description:</b>	Override the OCSP responder URI specified in the certificate's AIA extension
<b>Syntax:</b>	SSLStaplingForceURL <i>uri</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

This directive overrides the URI of an OCSP responder as obtained from the authorityInfoAccess (AIA) extension of the certificate. One potential use is when a proxy is used for retrieving OCSP queries.



## SSLStaplingResponderTimeout Directive

<b>Description:</b>	Timeout for OCSP stapling queries
<b>Syntax:</b>	SSLStaplingResponderTimeout <i>seconds</i>
<b>Default:</b>	SSLStaplingResponderTimeout 10
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

This option sets the timeout for queries to OCSP responders when [SSLUseStapling](#) is enabled and mod\_ssl is querying a responder for OCSP stapling purposes.



## SSLStaplingResponseMaxAge Directive

<b>Description:</b>	Maximum allowable age for OCSP stapling responses
<b>Syntax:</b>	SSLStaplingResponseMaxAge <i>seconds</i>
<b>Default:</b>	SSLStaplingResponseMaxAge -1
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

This option sets the maximum allowable age ("freshness") when considering OCSP responses for stapling purposes, i.e. when [SSLUseStapling](#) is turned on. The default value (-1) does not enforce a maximum age, which means that OCSP responses are considered valid as long as their nextUpdate field is in the future.



## SSLStaplingResponseTimeSkew Directive

<b>Description:</b>	Maximum allowable time skew for OCSP stapling response validation
<b>Syntax:</b>	SSLStaplingResponseTimeSkew <i>seconds</i>
<b>Default:</b>	SSLStaplingResponseTimeSkew 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

This option sets the maximum allowable time skew when mod\_ssl checks the thisUpdate and nextUpdate fields of OCSP responses which get included in the TLS handshake (OCSP stapling). Only applicable if [SSLUseStapling](#) is turned on.



## SSLStaplingReturnResponderErrors

<b>Description:</b>	Pass stapling related OCSP errors on to client
<b>Syntax:</b>	SSLStaplingReturnResponderErrors on off
<b>Default:</b>	SSLStaplingReturnResponderErrors on
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

When enabled, mod\_ssl will pass responses from unsuccessful stapling related OCSP queries (such as responses with an overall status other than "successful", responses with a certificate status other than "good", expired responses etc.) on to the client. If set to off, only responses indicating a certificate status of "good" will be included in the TLS handshake.



## SSLStaplingStandardCacheTimeout Directive

<b>Description:</b>	Number of seconds before expiring responses in the OCSP stapling cache
<b>Syntax:</b>	SSLStaplingStandardCacheTimeout <i>seconds</i>
<b>Default:</b>	SSLStaplingStandardCacheTimeout 3600
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

Sets the timeout in seconds before responses in the OCSP stapling cache (configured through [SSLStaplingCache](#)) will expire. This directive applies to *valid* responses, while [SSLStaplingErrorCacheTimeout](#) is used for controlling the timeout for invalid/unavailable responses.



<b>Description:</b>	Whether to allow non-SNI clients to access a name-based virtual host.
<b>Syntax:</b>	SSLStrictSNIVHostCheck on off
<b>Default:</b>	SSLStrictSNIVHostCheck off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in Apache 2.2.12 and later

This directive sets whether a non-SNI client is allowed to access a name-based virtual host. If set to on in the default name-based virtual host, clients that are SNI unaware will not be allowed to access *any* virtual host, belonging to this particular IP / port combination. If set to on in any other virtual host, SNI unaware clients are not allowed to access this particular virtual host.

This option is only available if httpd was compiled against an SNI capable version of OpenSSL.

### Example

```
SSLStrictSNIVHostCheck on
```



<b>Description:</b>	Variable name to determine user name
<b>Syntax:</b>	SSLUserName <i>varname</i>
<b>Context:</b>	server config, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the "user" field in the Apache request object. This is used by lower modules to identify the user with a character string. In particular, this may cause the environment variable REMOTE\_USER to be set. The *varname* can be any of the [SSL environment variables](#).

Note that this directive has no effect if the FakeBasicAuth option is used (see [SSLOptions](#)).

### Example

```
SSLUserName SSL_CLIENT_S_DN_CN
```



## OCSP Stapling Directive

<b>Description:</b>	Enable stapling of OCSP responses in the TLS handshake
<b>Syntax:</b>	SSLUseStapling on off
<b>Default:</b>	SSLUseStapling off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available if using OpenSSL 0.9.8h or later

This option enables OCSP stapling, as defined by the "Certificate Status Request" TLS extension specified in RFC 6066. If enabled (and requested by the client), mod\_ssl will include an OCSP response for its own certificate in the TLS handshake. Configuring an [SSLStaplingCache](#) is a prerequisite for enabling OCSP stapling.

OCSP stapling relieves the client of querying the OCSP responder on its own, but it should be noted that with the RFC 6066 specification, the server's CertificateStatus reply may only include an OCSP response for a single cert. For server certificates with intermediate CA certificates in their chain (the typical case nowadays), stapling in its current implementation therefore only partially achieves the stated goal of "saving roundtrips and resources" - see also [RFC 6961](#) (TLS Multiple Certificate Status Extension).

When OCSP stapling is enabled, the ssl-stapling mutex is used to control access to the OCSP stapling cache in order to prevent corruption, and the ssl-stapling-refresh mutex is used to control refreshes of OCSP responses. These mutexes can be configured using the [Mutex](#) directive.



<b>Description:</b>	Type of Client Certificate verification
<b>Syntax:</b>	SSLVerifyClient <i>level</i>
<b>Default:</b>	SSLVerifyClient none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the Certificate verification level for the Client Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification level after the HTTP request was read but before the HTTP response is sent.

The following levels are available for *level*:

- **none**: no client Certificate is required at all
- **optional**: the client *may* present a valid Certificate
- **require**: the client *has to* present a valid Certificate
- **optional\_no\_ca**: the client may present a valid Certificate but it need not to be (successfully) verifiable. This option cannot be relied upon for client authentication.

### Example

```
SSLVerifyClient require
```



<b>Description:</b>	Maximum depth of CA Certificates in Client Certificate verification
<b>Syntax:</b>	SSLVerifyDepth <i>number</i>
<b>Default:</b>	SSLVerifyDepth 1
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets how deeply mod\_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the client certificate. A depth of 0 means that self-signed client certificates are accepted only, the default depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under [SSLCACertificatePath](#)), etc.

### Example

```
SSLVerifyDepth 10
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_status

```
┆  
┆  
┆ Base  
┆ status_module  
┆ mod_status.c
```

Status . HTML .  
( ) .  
.  
:  

- worker
- (idle) worker
- worker , worker worker (\*)
- (\*)
- 
- , (\*)
- worker CPU (\*)
- (\*)

"(\*)" .



foo.com

httpd.conf

```
<Location /server-status>
  SetHandler server-status

  Order Deny,Allow
  Deny from all
  Allow from .foo.com
</Location>
```

<http://your.server.name/server-status>

.



---

```
"" status . N
http://your.server.name/server-status?refresh=N
```

.



status

```
http://your.server.name/server-status?auto
status . /support
log_server_status Perl .
```

**mod\_status**

( , .htaccess



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_substitute`

<b>Description:</b>	Perform search and replace operations on response bodies
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>substitute_module</code>
<b>Source File:</b>	<code>mod_substitute.c</code>
<b>Compatibility:</b>	Available in Apache HTTP Server 2.2.7 and later

### Summary

`mod_substitute` provides a mechanism to perform both regular expression and fixed string substitutions on response bodies.



<b>Description:</b>	Pattern to filter the response content
<b>Syntax:</b>	<code>Substitute</code> <code>s/pattern/substitution/[infq]</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_substitute

The `Substitute` directive specifies a search and replace pattern to apply to the response body.

The meaning of the pattern can be modified by using any combination of these flags:

**i**

Perform a case-insensitive match.

**n**

By default the pattern is treated as a regular expression. Using the `n` flag forces the pattern to be treated as a fixed string.

**f**

The `f` flag causes `mod_substitute` to flatten the result of a substitution allowing for later substitutions to take place on the boundary of this one. This is the default.

**q**

The `q` flag causes `mod_substitute` to not flatten the buckets after each substitution. This can result in much faster response and a decrease in memory utilization, but should only be used if there is no possibility that the result of one substitution will ever match a pattern or regex of a subsequent one.

## Example

```
<Location "/">
  AddOutputFilterByType SUBSTITUTE text/html
  Substitute "s/foo/bar/ni"
</Location>
```

If either the pattern or the substitution contain a slash character then an alternative delimiter should be used:

## Example of using an alternate delimiter

```
<Location "/">
  AddOutputFilterByType SUBSTITUTE text/html
  Substitute "s|<BR */?>|<br />|i"
</Location>
```

Backreferences can be used in the comparison and in the substitution, when regular expressions are used, as illustrated in the following example:

## Example of using backreferences and captures

```
<Location "/">
  AddOutputFilterByType SUBSTITUTE text/html
  # "foo=k,bar=k" -> "foo/bar=k"
  Substitute "s|foo=(\w+),bar=\1|foo/bar=$1"
</Location>
```

A common use scenario for `mod_substitute` is the situation in which a front-end server proxies requests to a back-end server which returns HTML with hard-coded embedded URLs that refer to the back-end server. These URLs don't work for the end-user, since the back-end server is unreachable.

In this case, `mod_substitute` can be used to rewrite those URLs into something that will work from the front end:

## Rewriting URLs embedded in proxied content

```
ProxyPass      "/blog/" "http://internal.blog.example.com"
ProxyPassReverse "/blog/" "http://internal.blog.example.com/"

Substitute "s|http://internal.blog.example.com/|http://www.examp
```

ProxyPassReverse modifies any Location (redirect) headers that are sent by the back-end server, and, in this example, Substitute takes care of the rest of the problem by fixing up the HTML response as well.



<b>Description:</b>	Change the merge order of inherited patterns
<b>Syntax:</b>	<code>SubstituteInheritBefore on off</code>
<b>Default:</b>	<code>SubstituteInheritBefore off</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_substitute
<b>Compatibility:</b>	Available in httpd 2.4.17 and later

Whether to apply the inherited `Substitute` patterns first (on), or after the ones of the current context (off).

`SubstituteInheritBefore` is itself inherited, hence contexts that inherit it (those that don't specify their own `SubstituteInheritBefore` value) will apply the closest defined merge order.



<b>Description:</b>	Set the maximum line size
<b>Syntax:</b>	<code>SubstituteMaxLineLength</code> <i>bytes(b B k K m M g G)</i>
<b>Default:</b>	<code>SubstituteMaxLineLength 1m</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_substitute
<b>Compatibility:</b>	Available in httpd 2.4.11 and later

The maximum line size handled by `mod_substitute` is limited to restrict memory use. The limit can be configured using `SubstituteMaxLineLength`. The value can be given as the number of bytes and can be suffixed with a single letter b, B, k, K, m, M, g, G to provide the size in bytes, kilobytes, megabytes or gigabytes respectively.

### Example

```
<Location "/">
    AddOutputFilterByType SUBSTITUTE text/html
    SubstituteMaxLineLength 10m
    Substitute "s/foo/bar/ni"
</Location>
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_suexec

- CGI
- Extension
- suexec\_module
- mod\_suexec.c
- 2.0

[suexec](#) CGI



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[SuEXEC](#)



## SuexecUserGroup

- [: CGI](#)
- [: SuexecUserGroup \*User Group\*](#)
- [: ,](#)
- [: Extension](#)
- [: mod\\_suexec](#)
- [: SuexecUserGroup 2.0 .](#)

**SuexecUserGroup CGI** . CGI  
User . 1.3 VirtualHost Us  
Group .

```
SuexecUserGroup nobody nogroup
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

## mod\_unique\_id

- 
- 
- Extension
- unique\_id\_module
- mod\_unique\_id.c

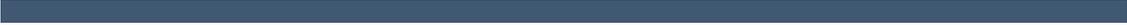
""

(identifier) .

UNIQUE\_ID

, .





. Windows NT .

, . httpd .

(cluster) . .

, . ( NTP

- NTP
- .

pid ( id) 32 . pid 32

httpd httpd

. IP httpd pid .

(timestamp, 1970 1 1 ) 16  
 . , 65536 . ( ip\_a  
 counter ) httpd 65536 . pid

httpd ( 10 ) 65536 . ( .)

( . ) .

(fork) pid , pid . (pid  
 32 .) pid . pid  
 . , 65536 .  
 32768 pid , .)

. , ( . ) . pid . ,

```

seed          rand() ,          s
.
?          500 (
.          .) .
.          500          . pid 50
          500          1000
          1.5% . ,
          () 32          .
"" .          (UTC),
. x86          . UTC
NTP          UTC .
          UNIQUE_ID MIME base64          112 (32 IP , 32
pid, 32 , 16 )          [A-Za-z0-9@-] . MIME
base64          [A-Za-z0-9+ /] + / URL .
.
,          ,          UNIQUE_ID .
          UNIQUE_ID          .
,          .
          (flag second) .
.
          . Windows NT
,          .
.          (          ↑
          ( pid ).
. ( , 32 IP          ,

```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module mod\_unixd

<b>Description:</b>	Basic (required) security for Unix-family platforms.
<b>Status:</b>	Base
<b>Module Identifier:</b>	unixd_module
<b>Source File:</b>	mod_unixd.c



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[suEXEC support](#)



<b>Description:</b>	Directory for apache to run <code>chroot(8)</code> after startup.
<b>Syntax:</b>	<code>ChrootDir /path/to/directory</code>
<b>Default:</b>	none
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_unixd</code>
<b>Compatibility:</b>	Available in Apache 2.2.10 and later

This directive tells the server to `chroot(8)` to the specified directory after startup, but before accepting requests over the 'net.

Note that running the server under `chroot` is not simple, and requires additional setup, particularly if you are running scripts such as CGI or PHP. Please make sure you are properly familiar with the operation of `chroot` before attempting to use this feature.



<b>Description:</b>	Group under which the server will answer requests
<b>Syntax:</b>	Group <i>unix-group</i>
<b>Default:</b>	Group #-1
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_unixd

The **Group** directive sets the group under which the server will answer requests. In order to use this directive, the server must be run initially as root. If you start the server as a non-root user, it will fail to change to the specified group, and will instead continue to run as the group of the original user. *Unix-group* is one of:

### A group name

Refers to the given group by name.

### # followed by a group number.

Refers to a group by its number.

### Example

```
Group www-group
```

It is recommended that you set up a new group specifically for running the server. Some admins use user nobody, but this is not always possible or desirable.

### Security

Don't set **Group** (or **User**) to root unless you know exactly what you are doing, and what the dangers are.

### See also

- VHostGroup
- SuexecUserGroup



<b>Description:</b>	Enable or disable the suEXEC feature
<b>Syntax:</b>	Suexec On Off
<b>Default:</b>	On if suexec binary exists with proper owner and mode, Off otherwise
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_unixd

When On, startup will fail if the suexec binary doesn't exist or has an invalid owner or file mode.

When Off, suEXEC will be disabled even if the suexec binary exists and has a valid owner and file mode.



**Description:** The userid under which the server will answer requests

**Syntax:** User *unix-userid*

**Default:** User #-1

**Context:** server config

**Status:** Base

**Module:** mod\_unixd

The **User** directive sets the user ID as which the server will answer requests. In order to use this directive, the server must be run initially as root. If you start the server as a non-root user, it will fail to change to the lesser privileged user, and will instead continue to run as that original user. If you do start the server as root, then it is normal for the parent process to remain running as root. *Unix-userid* is one of:

### **A username**

Refers to the given user by name.

### **# followed by a user number.**

Refers to a user by its number.

The user should have no privileges that result in it being able to access files that are not intended to be visible to the outside world, and similarly, the user should not be able to execute code that is not meant for HTTP requests. It is recommended that you set up a new user and group specifically for running the server. Some admins use user nobody, but this is not always desirable, since the nobody user can have other uses on the system.

### **Security**

Don't set **User** (or **Group**) to root unless you know exactly what you are doing, and what the dangers are.

---

## See also

- [VHostUser](#)
- [SuexecUserGroup](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

# mod\_userdir

```
┆  
┆ Base  
┆ userdir_module  
┆ mod_userdir.c
```

`http://example.com/~user/` .



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

[URL](#)

[public\\_html](#)



```

:
: UserDir directory-filename
: UserDir public_html
:
: ,
: Base
: mod_userdir

```

### UserDir

*Directory-filename* :

- .
- disabled . enabled ( ) -
- disabled . enabled
- enabled . disable  
disabled , .

Userdir enabled disabled ,  
 . http://www.foo.com/~bob/one/two.html  
 :

### UserDir

```

UserDir public_html ~bob/public_html/one/two.html
UserDir /usr/web /usr/web/bob/one/two.html
UserDir /home/*/www /home/bob/www/one/two.html

```

:

### UserDir

```

UserDir http://www.foo.com/users/bob/one/two.h
http://www.foo.com/users

```

```
UserDir http://www.foo.com/bob/usr/one/two.htm
http://www.foo.com/*/usr
UserDir http://www.foo.com/~bob/one/two.html
http://www.foo.com/~*/
```

---

```
; , "UserDir ./" "/~root"
"/" . " UserDir disabled root" .
Directory _ .
```

:

```
UserDir , :
```

```
UserDir disabled
UserDir enabled user1 user2 user3
```

```
UserDir , :
```

```
UserDir enabled
UserDir disabled user4 user5 user6
```

. :

```
Userdir public_html /usr/web http://www.foo.com/
```

```
http://www.foo.com/~bob/one/two.html ,
~bob/public_html/one/two.html , /usr/web/bob/one/two.html
, http://www.foo.com/bob/one/two.html .
```

```
. ,
.
```

- [public\\_html](#)





[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module mod\_usertrack

<b>Description:</b>	<i>Clickstream</i> logging of user activity on a site
<b>Status:</b>	Extension
<b>Module Identifier:</b>	usertrack_module
<b>Source File:</b>	mod_usertrack.c

### Summary

Provides tracking of a user through your website via browser cookies.



## Logging

[mod\\_usertrack](#) sets a cookie which can be logged via [mod\\_log\\_config](#) configurable logging formats:

```
LogFormat "%{Apache}n %r %t" usertrack  
CustomLog logs/clickstream.log usertrack
```



<b>Description:</b>	The domain to which the tracking cookie applies
<b>Syntax:</b>	CookieDomain <i>domain</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive controls the setting of the domain to which the tracking cookie applies. If not present, no domain is included in the cookie header field.

The domain string **must** begin with a dot, and **must** include at least one embedded dot. That is, `.example.com` is legal, but `www.example.com` and `.com` are not.

Most browsers in use today will not allow cookies to be set for a two-part top level domain, such as `.co.uk`, although such a domain ostensibly fulfills the requirements above. These domains are equivalent to top level domains such as `.com`, and allowing such cookies may be a security risk. Thus, if you are under a two-part top level domain, you should still use your actual domain, as you would with any other top level domain (for example `.example.co.uk`).

```
CookieDomain .example.com
```



<b>Description:</b>	Expiry time for the tracking cookie
<b>Syntax:</b>	<code>CookieExpires <i>expiry-period</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

When used, this directive sets an expiry time on the cookie generated by the usertrack module. The *expiry-period* can be given either as a number of seconds, or in the format such as "2 weeks 3 days 7 hours". Valid denominations are: years, months, weeks, days, hours, minutes and seconds. If the expiry time is in any format other than one number indicating the number of seconds, it must be enclosed by double quotes.

If this directive is not used, cookies last only for the current browser session.

```
CookieExpires "3 weeks"
```



<b>Description:</b>	Name of the tracking cookie
<b>Syntax:</b>	CookieName <i>token</i>
<b>Default:</b>	CookieName Apache
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive allows you to change the name of the cookie this module uses for its tracking purposes. By default the cookie is named "Apache".

You must specify a valid cookie name; results are unpredictable if you use a name containing unusual characters. Valid characters include A-Z, a-z, 0-9, "\_", and "-".

```
CookieName clicktrack
```



<b>Description:</b>	Format of the cookie header field
<b>Syntax:</b>	CookieStyle <i>Netscape   Cookie   Cookie2   RFC2109   RFC2965</i>
<b>Default:</b>	CookieStyle Netscape
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive controls the format of the cookie header field. The three formats allowed are:

- **Netscape**, which is the original but now deprecated syntax. This is the default, and the syntax Apache has historically used.
- **Cookie** or **RFC2109**, which is the syntax that superseded the Netscape syntax.
- **Cookie2** or **RFC2965**, which is the most current cookie syntax.

Not all clients can understand all of these formats, but you should use the newest one that is generally acceptable to your users' browsers. At the time of writing, most browsers support all three of these formats, with Cookie2 being the preferred format.

```
CookieStyle Cookie2
```



## CookieTracking Directive

<b>Description:</b>	Enables tracking cookie
<b>Syntax:</b>	CookieTracking on off
<b>Default:</b>	CookieTracking off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

When [mod\\_usertrack](#) is loaded, and `CookieTracking on` is set, Apache will send a user-tracking cookie for all new requests. This directive can be used to turn this behavior on or off on a per-server or per-directory basis. By default, enabling [mod\\_usertrack](#) will **not** activate cookies.

```
CookieTracking on
```



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >



## IFVERSION

```

:
:      <IfVersion [[!]operator] version> ...
:      </IfVersion>
:      , , directory, .htaccess
Override : All
:      Extension
:      mod_version

```

```

<IfVersion>
version      2.1.0 2.2  major[.minor[.patch]].
patch .      0 .      ope

```

<b>operator</b>	
= ==	
>	
>=	
<	
<=	

```

<IfVersion >= 2.1>
# 2.1.0
# .
</IfVersion>

```

. .



<i>operator</i>	
=    ==	<i>version / regex/</i>
~	<i>version regex</i>

```
<IfVersion = /^2.1.[01234]$/>
# ,
</IfVersion>
```

( ! ) .

```
<IfVersion !~ ^2.1.[01234]$>
#
</IfVersion>
```

*operator* = .



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_vhost_alias`

<b>Description:</b>	Provides for dynamically configured mass virtual hosting
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>vhost_alias_module</code>
<b>Source File:</b>	<code>mod_vhost_alias.c</code>

## Summary

This module creates dynamically configured virtual hosts, by allowing the IP address and/or the `Host :` header of the HTTP request to be used as part of the pathname to determine what files to serve. This allows for easy use of a huge number of virtual hosts with similar configurations.

### Note

If `mod_alias` or `mod_userdir` are used for translating URIs to filenames, they will override the directives of `mod_vhost_alias` described below. For example, the following configuration will map `/cgi-bin/script.pl` to `/usr/local/apache2/cgi-bin/script.pl` in all cases:

```
ScriptAlias "/cgi-bin/" "/usr/local/apache2/cgi-bin/"
VirtualScriptAlias "/never/found/%0/cgi-bin/"
```



## Bugfix checklist

[httpd changelog](#)

[Known issues](#)

[Report a bug](#)

## See also

[UseCanonicalName](#)

[Dynamically configured mass virtual hosting](#)



## Directory Name Interpolation

All the directives in this module interpolate a string into a pathname. The interpolated string (henceforth called the "name") may be either the server name (see the [UseCanonicalName](#) directive for details on how this is determined) or the IP address of the virtual host on the server in dotted-quad format. The interpolation is controlled by specifiers inspired by `printf` which have a number of formats:

<code>%%</code>	insert a %
<code>%p</code>	insert the port number of the virtual host
<code>%N.M</code>	insert (part of) the name

N and M are used to specify substrings of the name. N selects from the dot-separated components of the name, and M selects characters within whatever N has selected. M is optional and defaults to zero if it isn't present; the dot must be present if and only if M is present. The interpretation is as follows:

<code>0</code>	the whole name
<code>1</code>	the first part
<code>2</code>	the second part
<code>-1</code>	the last part
<code>-2</code>	the penultimate part
<code>2+</code>	the second and all subsequent parts
<code>-2+</code>	the penultimate and all preceding parts
<code>1+</code> and <code>-1+</code>	the same as <code>0</code>

If N or M is greater than the number of parts available a single underscore is interpolated.



For simple name-based virtual hosts you might use the following directives in your server configuration file:

```
UseCanonicalName    Off
VirtualDocumentRoot "/usr/local/apache/vhost
```

A request for  
`http://www.example.com/directory/file.html` will be  
satisfied by the file  
`/usr/local/apache/vhosts/www.example.com/directory`

For a very large number of virtual hosts it is a good idea to arrange the files to reduce the size of the vhosts directory. To do this you might use the following in your configuration file:

```
UseCanonicalName    Off
VirtualDocumentRoot "/usr/local/apache/vhost
```

A request for  
`http://www.domain.example.com/directory/file.html`  
will be satisfied by the file  
`/usr/local/apache/vhosts/example.com/d/o/m/domain`

A more even spread of files can be achieved by hashing from the end of the name, for example:

```
VirtualDocumentRoot "/usr/local/apache/vhost
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/n/i/a/domain`

Alternatively you might use:

```
VirtualDocumentRoot "/usr/local/apache/vhost
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/d/o/m/a/i/n/di`

A very common request by users is the ability to point multiple domains to multiple document roots without having to worry about the length or number of parts of the hostname being requested. If the requested hostname is `sub.www.domain.example.com` instead of simply `www.domain.example.com`, then using `%3+` will result in the document root being `/usr/local/apache/vhosts/domain.example.com/...` instead of the intended `example.com` directory. In such cases, it can be beneficial to use the combination `%-2.0.%-1.0`, which will always yield the domain name and the tld, for example `example.com` regardless of the number of subdomains appended to the hostname. As such, one can make a configuration that will direct all first, second or third level subdomains to the same directory:

```
VirtualDocumentRoot "/usr/local/apache/vhost
```

In the example above, both `www.example.com` as well as `www.sub.example.com` or `example.com` will all point to `/usr/local/apache/vhosts/example.com`.

For IP-based virtual hosting you might use the following in your configuration file:

```
UseCanonicalName DNS
```

```
VirtualDocumentRootIP "/usr/local/apache/vho
VirtualScriptAliasIP  "/usr/local/apache/vho
```

A request for  
`http://www.domain.example.com/directory/file.html`  
would be satisfied by the file  
`/usr/local/apache/vhosts/10/20/30/40/docs/directo`  
if the IP address of `www.domain.example.com` were  
10.20.30.40. A request for  
`http://www.domain.example.com/cgi-bin/script.pl`  
would be satisfied by executing the program  
`/usr/local/apache/vhosts/10/20/30/40/cgi-`  
`bin/script.pl`.

If you want to include the `.` character in a  
`VirtualDocumentRoot` directive, but it clashes with a `%`  
directive, you can work around the problem in the following way:

```
VirtualDocumentRoot "/usr/local/apache/vhost
```

A request for  
`http://www.domain.example.com/directory/file.html`  
will be satisfied by the file  
`/usr/local/apache/vhosts/domain.example/directory.`

The [LogFormat](#) directives `%V` and `%A` are useful in conjunction  
with this module.



<b>Description:</b>	Dynamically configure the location of the document root for a given virtual host
<b>Syntax:</b>	<code>VirtualDocumentRoot</code> <i>interpolated-directory</i>  none
<b>Default:</b>	<code>VirtualDocumentRoot</code> none
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_vhost_alias

The `VirtualDocumentRoot` directive allows you to determine where Apache HTTP Server will find your documents based on the value of the server name. The result of expanding *interpolated-directory* is used as the root of the document tree in a similar manner to the `DocumentRoot` directive's argument. If *interpolated-directory* is none then `VirtualDocumentRoot` is turned off. This directive cannot be used in the same context as `VirtualDocumentRootIP`.

#### Note

`VirtualDocumentRoot` will override any `DocumentRoot` directives you may have put in the same context or child contexts. Putting a `VirtualDocumentRoot` in the global server scope will effectively override `DocumentRoot` directives in any virtual hosts defined later on, unless you set `VirtualDocumentRoot` to None in each virtual host.



<b>Description:</b>	Dynamically configure the location of the document root for a given virtual host
<b>Syntax:</b>	<code>VirtualDocumentRootIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualDocumentRootIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualDocumentRootIP` directive is like the `VirtualDocumentRoot` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.



## VirtualScriptAlias Directive

<b>Description:</b>	Dynamically configure the location of the CGI directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAlias <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAlias none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualScriptAlias` directive allows you to determine where Apache httpd will find CGI scripts in a similar manner to `VirtualDocumentRoot` does for other documents. It matches requests for URIs starting `/cgi-bin/`, much like `ScriptAlias /cgi-bin/` would.



<b>Description:</b>	Dynamically configure the location of the CGI directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAliasIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAliasIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualScriptAliasIP` directive is like the `VirtualScriptAlias` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

# Apache Module `mod_watchdog`

<b>Description:</b>	provides infrastructure for other modules to periodically run tasks
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>watchdog_module</code>
<b>Source File:</b>	<code>mod_watchdog.c</code>
<b>Compatibility:</b>	Available in Apache 2.3 and later

## Summary

`mod_watchdog` defines programmatic hooks for other modules to periodically run tasks. These modules can register handlers for `mod_watchdog` hooks. Currently, the following modules in the Apache distribution use this functionality:

- `mod_heartbeat`
- `mod_heartmonitor`

To allow a module to use `mod_watchdog` functionality, `mod_watchdog` itself must be statically linked to the server core or, if a dynamic module, be loaded before the calling module.



## WatchdogInterval Directive

<b>Description:</b>	Watchdog interval in seconds
<b>Syntax:</b>	<code>WatchdogInterval</code> <i>number-of-seconds</i>
<b>Default:</b>	<code>WatchdogInterval</code> 1
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_watchdog</code>

Sets the interval at which the `watchdog_step` hook runs. Default is to run every second.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Modules](#)

## Apache Module `mod_xml2enc`

<b>Description:</b>	Enhanced charset/internationalisation support for libxml2-based filter modules
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>xml2enc_module</code>
<b>Source File:</b>	<code>mod_xml2enc.c</code>
<b>Compatibility:</b>	Version 2.4 and later. Available as a third-party module for 2.2.x versions

### Summary

This module provides enhanced internationalisation support for markup-aware filter modules such as `mod_proxy_html`. It can automatically detect the encoding of input data and ensure they are correctly processed by the `libxml2` parser, including converting to Unicode (UTF-8) where necessary. It can also convert data to an encoding of choice after markup processing, and will ensure the correct *charset* value is set in the HTTP *Content-Type* header.



There are two usage scenarios: with modules programmed to work with `mod_xml2enc`, and with those that are not aware of it:

### Filter modules enabled for `mod_xml2enc`

Modules such as `mod_proxy_html` version 3.1 and up use the `xml2enc_charset` optional function to retrieve the charset argument to pass to the libxml2 parser, and may use the `xml2enc_filter` optional function to postprocess to another encoding. Using `mod_xml2enc` with an enabled module, no configuration is necessary: the other module will configure `mod_xml2enc` for you (though you may still want to customise it using the configuration directives below).

### Non-enabled modules

To use it with a libxml2-based module that isn't explicitly enabled for `mod_xml2enc`, you will have to configure the filter chain yourself. So to use it with a filter `foo` provided by a module `mod_foo` to improve the latter's i18n support with HTML and XML, you could use

```
FilterProvider iconv      xml2enc Content-Ty
FilterProvider iconv      xml2enc Content-Ty
FilterProvider markup     foo Content-Type $
FilterProvider markup     foo Content-Type $
FilterChain      iconv markup
```

`mod_foo` will now support any character set supported by either (or both) of libxml2 or `apr_xlate/iconv`.



## Programming API

Programmers writing libxml2-based filter modules are encouraged to enable them for `mod_xml2enc`, to provide strong i18n support for your users without reinventing the wheel. The programming API is exposed in `mod_xml2enc.h`, and a usage example is [mod\\_proxy.html](#).



## Detecting an Encoding

Unlike `mod_charset_lite`, `mod_xml2enc` is designed to work with data whose encoding cannot be known in advance and thus configured. It therefore uses 'sniffing' techniques to detect the encoding of HTTP data as follows:

1. If the HTTP *Content-Type* header includes a *charset* parameter, that is used.
2. If the data start with an XML Byte Order Mark (BOM) or an XML encoding declaration, that is used.
3. If an encoding is declared in an HTML `<META>` element, that is used.
4. If none of the above match, the default value set by `xml2EncDefault` is used.

The rules are applied in order. As soon as a match is found, it is used and detection is stopped.



## Output Encoding

[libxml2](#) always uses UTF-8 (Unicode) internally, and libxml2-based filter modules will output that by default. `mod_xml2enc` can change the output encoding through the API, but there is currently no way to configure that directly.

Changing the output encoding should (in theory, at least) never be necessary, and is not recommended due to the extra processing load on the server of an unnecessary conversion.



## Unsupported Encodings

If you are working with encodings that are not supported by any of the conversion methods available on your platform, you can still alias them to a supported encoding using `xml2EncAlias`.



<b>Description:</b>	Recognise Aliases for encoding values
<b>Syntax:</b>	<code>xml2EncAlias charset alias [alias ...]</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_xml2enc

This server-wide directive aliases one or more encoding to another encoding. This enables encodings not recognised by libxml2 to be handled internally by libxml2's encoding support using the translation table for a recognised encoding. This serves two purposes: to support character sets (or names) not recognised either by libxml2 or iconv, and to skip conversion for an encoding where it is known to be unnecessary.



<b>Description:</b>	Sets a default encoding to assume when absolutely no information can be <a href="#">automatically detected</a>
<b>Syntax:</b>	<code>xml2EncDefault</code> <i>name</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Base
<b>Module:</b>	mod_xml2enc
<b>Compatibility:</b>	Version 2.4.0 and later; available as a third-party module for earlier versions.

If you are processing data with known encoding but no encoding information, you can set this default to help mod\_xml2enc process the data correctly. For example, to work with the default value of Latin1 (*iso-8859-1* specified in HTTP/1.0, use

```
xml2EncDefault iso-8859-1
```



<b>Description:</b>	Advise the parser to skip leading junk.
<b>Syntax:</b>	<code>xml2StartParse <i>element</i> [<i>element</i> ...]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Base
<b>Module:</b>	mod_xml2enc

Specify that the markup parser should start at the first instance of any of the elements specified. This can be used as a workaround where a broken backend inserts leading junk that messes up the parser ([example here](#)).

It should never be used for XML, nor well-formed HTML.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

# Apache 1.3 API notes

## Warning

This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

These are some notes on the Apache API and the data structures you have to deal with, *etc.* They are not yet nearly complete, but hopefully, they will help you get your bearings. Keep in mind that the API is still subject to change as we gain experience with it. (See the TODO file for what *might* be coming). However, it will be easy to adapt modules to any changes that are made. (We have more modules to adapt than you do).

A few notes on general pedagogical style here. In the interest of conciseness, all structure declarations here are incomplete -- the real ones have more slots that I'm not telling you about. For the most part, these are reserved to one component of the server core or another, and should be altered by modules with caution. However, in some cases, they really are things I just haven't gotten around to yet. Welcome to the bleeding edge.

Finally, here's an outline, to give you some bare idea of what's coming up, and in what order:

- [Basic concepts.](#)
  - [Handlers, Modules, and Requests](#)
  - [A brief tour of a module](#)
- [How handlers work](#)
  - [A brief tour of the request\\_rec](#)
  - [Where request\\_rec structures come from](#)

- [Handling requests, declining, and returning error codes](#)
- [Special considerations for response handlers](#)
- [Special considerations for authentication handlers](#)
- [Special considerations for logging handlers](#)
  
- [Resource allocation and resource pools](#)
- [Configuration, commands and the like](#)
  - [Per-directory configuration structures](#)
  - [Command handling](#)
  - [Side notes --- per-server configuration, virtual servers, etc.](#)



## Basic Concepts

We begin with an overview of the basic concepts behind the API, and how they are manifested in the code.

### Handlers, Modules, and Requests

Apache breaks down request handling into a series of steps, more or less the same way the Netscape server API does (although this API has a few more stages than NetSite does, as hooks for stuff I thought might be useful in the future). These are:

- URI -> Filename translation
- Auth ID checking [is the user who they say they are?]
- Auth access checking [is the user authorized *here*?]
- Access checking other than auth
- Determining MIME type of the object requested
- `Fixups' -- there aren't any of these yet, but the phase is intended as a hook for possible extensions like [SetEnv](#), which don't really fit well elsewhere.
- Actually sending a response back to the client.
- Logging the request

These phases are handled by looking at each of a succession of *modules*, looking to see if each of them has a handler for the phase, and attempting invoking it if so. The handler can typically do one of three things:

- *Handle* the request, and indicate that it has done so by returning the magic constant OK.
- *Decline* to handle the request, by returning the magic integer constant DECLINED. In this case, the server behaves in all respects as if the handler simply hadn't been there.
- Signal an error, by returning one of the HTTP error codes. This terminates normal handling of the request, although an `ErrorDocument` may be invoked to try to mop up, and it will be

logged in any case.

Most phases are terminated by the first module that handles them; however, for logging, `fixups', and non-access authentication checking, all handlers always run (barring an error). Also, the response phase is unique in that modules may declare multiple handlers for it, via a dispatch table keyed on the MIME type of the requested object. Modules may declare a response-phase handler which can handle *any* request, by giving it the key `*/*` (*i.e.*, a wildcard MIME type specification). However, wildcard handlers are only invoked if the server has already tried and failed to find a more specific response handler for the MIME type of the requested object (either none existed, or they all declined).

The handlers themselves are functions of one argument (a `request_rec` structure. *vide infra*), which returns an integer, as above.

## **A brief tour of a module**

At this point, we need to explain the structure of a module. Our candidate will be one of the messier ones, the CGI module -- this handles both CGI scripts and the [ScriptAlias](#) config file command. It's actually a great deal more complicated than most modules, but if we're going to have only one example, it might as well be the one with its fingers in every place.

Let's begin with handlers. In order to handle the CGI scripts, the module declares a response handler for them. Because of [ScriptAlias](#), it also has handlers for the name translation phase (to recognize [ScriptAliased](#) URIs), the type-checking phase (any [ScriptAliased](#) request is typed as a CGI script).

The module needs to maintain some per (virtual) server information, namely, the [ScriptAliases](#) in effect; the module

structure therefore contains pointers to a functions which builds these structures, and to another which combines two of them (in case the main server and a virtual server both have [ScriptAliases](#) declared).

Finally, this module contains code to handle the [ScriptAlias](#) command itself. This particular module only declares one command, but there could be more, so modules have *command tables* which declare their commands, and describe where they are permitted, and how they are to be invoked.

A final note on the declared types of the arguments of some of these commands: a *pool* is a pointer to a *resource pool* structure; these are used by the server to keep track of the memory which has been allocated, files opened, *etc.*, either to service a particular request, or to handle the process of configuring itself. That way, when the request is over (or, for the configuration pool, when the server is restarting), the memory can be freed, and the files closed, *en masse*, without anyone having to write explicit code to track them all down and dispose of them. Also, a *cmd\_parms* structure contains various information about the config file being read, and other status information, which is sometimes of use to the function which processes a config-file command (such as [ScriptAlias](#)). With no further ado, the module itself:

```
/* Declarations of handlers. */

int translate_scriptalias (request_rec *);
int type_scriptalias (request_rec *);
int cgi_handler (request_rec *);

/* Subsidiary dispatch table for response-phase
 * handlers, by MIME type */

handler_rec cgi_handlers[] = {
    { "application/x-httpd-cgi", cgi_handler },
    { NULL }
};
```

```

/* Declarations of routines to manipulate the
 * module's configuration info. Note that these are
 * returned, and passed in, as void *'s; the server
 * core keeps track of them, but it doesn't, and can't,
 * know their internal structure.
 */

void *make_cgi_server_config (pool *);
void *merge_cgi_server_config (pool *, void *, void *);

/* Declarations of routines to handle config-file commands */

extern char *script_alias(cmd_parms *, void *per_dir_config,
char *fake, char *real);

command_rec cgi_cmds[] = {
    { "ScriptAlias", script_alias, NULL, RSRC_CONF, TAKE2,
      "a fakename and a realname"},
    { NULL }
};

module cgi_module = {
    STANDARD_MODULE_STUFF,
    NULL, /* initializer */
    NULL, /* dir config creator */
    NULL, /* dir merger */
    make_cgi_server_config, /* server config */
    merge_cgi_server_config, /* merge server config */
    cgi_cmds, /* command table */
    cgi_handlers, /* handlers */
    translate_scriptalias, /* filename translation */
    NULL, /* check_user_id */
    NULL, /* check_auth */
    NULL, /* check_access */
    type_scriptalias, /* type_checker */
    NULL, /* fixups */
    NULL, /* logger */
    NULL /* header parser */
};

```



The sole argument to handlers is a `request_rec` structure. This structure describes a particular request which has been made to the server, on behalf of a client. In most cases, each connection to the client generates only one `request_rec` structure.

## A brief tour of the `request_rec`

The `request_rec` contains pointers to a resource pool which will be cleared when the server is finished handling the request; to structures containing per-server and per-connection information, and most importantly, information on the request itself.

The most important such information is a small set of character strings describing attributes of the object being requested, including its URI, filename, content-type and content-encoding (these being filled in by the translation and type-check handlers which handle the request, respectively).

Other commonly used data items are tables giving the MIME headers on the client's original request, MIME headers to be sent back with the response (which modules can add to at will), and environment variables for any subprocesses which are spawned off in the course of servicing the request. These tables are manipulated using the `ap_table_get` and `ap_table_set` routines.

Note that the Content - type header value *cannot* be set by module content-handlers using the `ap_table_*`( ) routines. Rather, it is set by pointing the `content_type` field in the `request_rec` structure to an appropriate string. *e.g.*,

```
r->content_type = "text/html";
```

Finally, there are pointers to two data structures which, in turn, point to per-module configuration structures. Specifically, these hold pointers to the data structures which the module has built to describe the way it has been configured to operate in a given directory (via `.htaccess` files or [<Directory>](#) sections), for private data it has built in the course of servicing the request (so modules' handlers for one phase can pass `notes' to their handlers for other phases). There is another such configuration vector in the `server_rec` data structure pointed to by the `request_rec`, which contains per (virtual) server configuration data.

Here is an abridged declaration, giving the fields most commonly used:

```
struct request_rec {

    pool *pool;
    conn_rec *connection;
    server_rec *server;

    /* What object is being requested */

    char *uri;
    char *filename;
    char *path_info;
    char *args;          /* QUERY_ARGS, if any */
    struct stat finfo;   /* Set by server core;
                        * st_mode set to zero if no such file */

    char *content_type;
    char *content_encoding;

    /* MIME header environments, in and out. Also,
     * an array containing environment variables to
     * be passed to subprocesses, so people can write
     * modules to add to that environment.
     *
     * The difference between headers_out and
     * err_headers_out is that the latter are printed
     * even on error, and persist across internal
     * redirects (so the headers printed for
     * ErrorDocument handlers will have them).
```

```

*/

table *headers_in;
table *headers_out;
table *err_headers_out;
table *subprocess_env;

/* Info about the request itself... */

int header_only;      /* HEAD request, as opposed to GET */
char *protocol;      /* Protocol, as given to us, or HTTP/0.9 */
char *method;        /* GET, HEAD, POST, etc. */
int method_number;   /* M_GET, M_POST, etc. */
/* Info for logging */

char *the_request;
int bytes_sent;

/* A flag which modules can set, to indicate that
 * the data being returned is volatile, and clients
 * should be told not to cache it.
 */

int no_cache;

/* Various other config info which may change
 * with .htaccess files
 * These are config vectors, with one void*
 * pointer for each module (the thing pointed
 * to being the module's business).
 */

void *per_dir_config; /* Options set in config files, etc. */
void *request_config; /* Notes on *this* request */
};

```

## Where request\_rec structures come from

Most request\_rec structures are built by reading an HTTP request from a client, and filling in the fields. However, there are a few exceptions:

- If the request is to an imagemap, a type map (*i.e.*, a \*.var

file), or a CGI script which returned a local `Location:', then the resource which the user requested is going to be ultimately located by some URI other than what the client originally supplied. In this case, the server does an *internal redirect*, constructing a new `request_rec` for the new URI, and processing it almost exactly as if the client had requested the new URI directly.

- If some handler signaled an error, and an `ErrorDocument` is in scope, the same internal redirect machinery comes into play.
- Finally, a handler occasionally needs to investigate `what would happen if' some other request were run. For instance, the directory indexing module needs to know what MIME type would be assigned to a request for each directory entry, in order to figure out what icon to use.

Such handlers can construct a *sub-request*, using the functions `ap_sub_req_lookup_file`, `ap_sub_req_lookup_uri`, and `ap_sub_req_method_uri`; these construct a new `request_rec` structure and processes it as you would expect, up to but not including the point of actually sending a response. (These functions skip over the access checks if the sub-request is for a file in the same directory as the original request).

(Server-side includes work by building sub-requests and then actually invoking the response handler for them, via the function `ap_run_sub_req`).

## **Handling requests, declining, and returning error codes**

As discussed above, each handler, when invoked to handle a

particular `request_rec`, has to return an `int` to indicate what happened. That can either be

- OK -- the request was handled successfully. This may or may not terminate the phase.
- DECLINED -- no erroneous condition exists, but the module declines to handle the phase; the server tries to find another.
- an HTTP error code, which aborts handling of the request.

Note that if the error code returned is REDIRECT, then the module should put a `Location` in the request's `headers_out`, to indicate where the client should be redirected to.

## Special considerations for response handlers

Handlers for most phases do their work by simply setting a few fields in the `request_rec` structure (or, in the case of access checkers, simply by returning the correct error code). However, response handlers have to actually send a request back to the client.

They should begin by sending an HTTP response header, using the function `ap_send_http_header`. (You don't have to do anything special to skip sending the header for HTTP/0.9 requests; the function figures out on its own that it shouldn't do anything). If the request is marked `header_only`, that's all they should do; they should return after that, without attempting any further output.

Otherwise, they should produce a request body which responds to the client as appropriate. The primitives for this are `ap_rputc` and `ap_rprintf`, for internally generated output, and `ap_send_fd`, to copy the contents of some `FILE *` straight to the client.

At this point, you should more or less understand the following

piece of code, which is the handler which handles GET requests which have no more specific handler; it also shows how conditional GETs can be handled, if it's desirable to do so in a particular response handler -- `ap_set_last_modified` checks against the `If-modified-since` value supplied by the client, if any, and returns an appropriate code (which will, if nonzero, be `USE_LOCAL_COPY`). No similar considerations apply for `ap_set_content_length`, but it returns an error code for symmetry.

```
int default_handler (request_rec *r)
{
    int errstatus;
    FILE *f;

    if (r->method_number != M_GET) return DECLINED;
    if (r->finfo.st_mode == 0) return NOT_FOUND;

    if ((errstatus = ap_set_content_length (r, r-
>finfo.st_size))
        || (errstatus = ap_set_last_modified (r, r-
>finfo.st_mtime)))
        return errstatus;

    f = fopen (r->filename, "r");

    if (f == NULL) {
        log_reason("file permissions deny server access", r-
>filename, r);
        return FORBIDDEN;
    }

    register_timeout ("send", r);
    ap_send_http_header (r);

    if (!r->header_only) send_fd (f, r);
    ap_pfclose (r->pool, f);
    return OK;
}
```

Finally, if all of this is too much of a challenge, there are a few ways out of it. First off, as shown above, a response handler which

has not yet produced any output can simply return an error code, in which case the server will automatically produce an error response. Secondly, it can punt to some other handler by invoking `ap_internal_redirect`, which is how the internal redirection machinery discussed above is invoked. A response handler which has internally redirected should always return OK.

(Invoking `ap_internal_redirect` from handlers which are *not* response handlers will lead to serious confusion).

## Special considerations for authentication handlers

Stuff that should be discussed here in detail:

- Authentication-phase handlers not invoked unless auth is configured for the directory.
- Common auth configuration stored in the core per-dir configuration; it has accessors `ap_auth_type`, `ap_auth_name`, and `ap_requires`.
- Common routines, to handle the protocol end of things, at least for HTTP basic authentication (`ap_get_basic_auth_pw`, which sets the `connection->user` structure field automatically, and `ap_note_basic_auth_failure`, which arranges for the proper `WWW-Authenticate:` header to be sent back).

## Special considerations for logging handlers

When a request has internally redirected, there is the question of what to log. Apache handles this by bundling the entire chain of redirects into a list of `request_rec` structures which are threaded through the `r->prev` and `r->next` pointers. The `request_rec` which is passed to the logging handlers in such cases is the one which was originally built for the initial request from the client; note that the `bytes_sent` field will only be correct in the last request in

the chain (the one for which a response was actually sent).



---

One of the problems of writing and designing a server-pool server is that of preventing leakage, that is, allocating resources (memory, open files, *etc.*), without subsequently releasing them. The resource pool machinery is designed to make it easy to prevent this from happening, by allowing resource to be allocated in such a way that they are *automatically* released when the server is done with them.

The way this works is as follows: the memory which is allocated, file opened, *etc.*, to deal with a particular request are tied to a *resource pool* which is allocated for the request. The pool is a data structure which itself tracks the resources in question.

When the request has been processed, the pool is *cleared*. At that point, all the memory associated with it is released for reuse, all files associated with it are closed, and any other clean-up functions which are associated with the pool are run. When this is over, we can be confident that all the resource tied to the pool have been released, and that none of them have leaked.

Server restarts, and allocation of memory and resources for per-server configuration, are handled in a similar way. There is a *configuration pool*, which keeps track of resources which were allocated while reading the server configuration files, and handling the commands therein (for instance, the memory that was allocated for per-server module configuration, log files and other files that were opened, and so forth). When the server restarts, and has to reread the configuration files, the configuration pool is cleared, and so the memory and file descriptors which were taken up by reading them the last time are made available for reuse.

It should be noted that use of the pool machinery isn't generally obligatory, except for situations like logging handlers, where you really need to register cleanups to make sure that the log file gets

closed when the server restarts (this is most easily done by using the function `ap_pfopen`, which also arranges for the underlying file descriptor to be closed before any child processes, such as for CGI scripts, are execed), or in case you are using the timeout machinery (which isn't yet even documented here). However, there are two benefits to using it: resources allocated to a pool never leak (even if you allocate a scratch string, and just forget about it); also, for memory allocation, `ap_palloc` is generally faster than `malloc`.

We begin here by describing how memory is allocated to pools, and then discuss how other resources are tracked by the resource pool machinery.

## Allocation of memory in pools

Memory is allocated to pools by calling the function `ap_palloc`, which takes two arguments, one being a pointer to a resource pool structure, and the other being the amount of memory to allocate (in chars). Within handlers for handling requests, the most common way of getting a resource pool structure is by looking at the `pool` slot of the relevant `request_rec`; hence the repeated appearance of the following idiom in module code:

```
int my_handler(request_rec *r)
{
    struct my_structure *foo;
    ...

    foo = (foo *)ap_palloc (r->pool, sizeof(my_structure));
}
```

Note that *there is no `ap_pfree`* -- `ap_palloc`d memory is freed only when the associated resource pool is cleared. This means that `ap_palloc` does not have to do as much accounting as `malloc()`; all it does in the typical case is to round up the size,

bump a pointer, and do a range check.

(It also raises the possibility that heavy use of `ap_palloc` could cause a server process to grow excessively large. There are two ways to deal with this, which are dealt with below; briefly, you can use `malloc`, and try to be sure that all of the memory gets explicitly freed, or you can allocate a sub-pool of the main pool, allocate your memory in the sub-pool, and clear it out periodically. The latter technique is discussed in the section on sub-pools below, and is used in the directory-indexing code, in order to avoid excessive storage allocation when listing directories with thousands of files).

## Allocating initialized memory

There are functions which allocate initialized memory, and are frequently useful. The function `ap_pcalloc` has the same interface as `ap_palloc`, but clears out the memory it allocates before it returns it. The function `ap_pstrdup` takes a resource pool and a `char *` as arguments, and allocates memory for a copy of the string the pointer points to, returning a pointer to the copy. Finally `ap_pstrcat` is a `varargs`-style function, which takes a pointer to a resource pool, and at least two `char *` arguments, the last of which must be `NULL`. It allocates enough memory to fit copies of each of the strings, as a unit; for instance:

```
ap_pstrcat (r->pool, "foo", "/", "bar", NULL);
```

returns a pointer to 8 bytes worth of memory, initialized to "foo/bar".

## Commonly-used pools in the Apache Web server

A pool is really defined by its lifetime more than anything else.

There are some static pools in `http_main` which are passed to various non-`http_main` functions as arguments at opportune times. Here they are:

### **permanent\_pool**

never passed to anything else, this is the ancestor of all pools

### **pconf**

- subpool of `permanent_pool`
- created at the beginning of a config "cycle"; exists until the server is terminated or restarts; passed to all config-time routines, either via `cmd->pool`, or as the "pool \*p" argument on those which don't take pools
- passed to the module `init()` functions

### **ptemp**

- sorry I lie, this pool isn't called this currently in 1.3, I renamed it this in my `pthreads` development. I'm referring to the use of `ptrans` in the parent... contrast this with the later definition of `ptrans` in the child.
- subpool of `permanent_pool`
- created at the beginning of a config "cycle"; exists until the end of config parsing; passed to config-time routines *via* `cmd->temp_pool`. Somewhat of a "bastard child" because it isn't available everywhere. Used for temporary scratch space which may be needed by some config routines but which is deleted at the end of config.

### **pchild**

- subpool of `permanent_pool`
- created when a child is spawned (or a thread is created); lives until that child (thread) is destroyed
- passed to the module `child_init` functions
- destruction happens right after the `child_exit` functions are called... (which may explain why I think `child_exit` is

redundant and unneeded)

### **ptrans**

- should be a subpool of pchild, but currently is a subpool of permanent\_pool, see above
- cleared by the child before going into the accept() loop to receive a connection
- used as connection->pool

### **r->pool**

- for the main request this is a subpool of connection->pool; for subrequests it is a subpool of the parent request's pool.
- exists until the end of the request (*i.e.*, ap\_destroy\_sub\_req, or in child\_main after process\_request has finished)
- note that r itself is allocated from r->pool; *i.e.*, r->pool is first created and then r is the first thing palloc()d from it

For almost everything folks do, r->pool is the pool to use. But you can see how other lifetimes, such as pchild, are useful to some modules... such as modules that need to open a database connection once per child, and wish to clean it up when the child dies.

You can also see how some bugs have manifested themselves, such as setting connection->user to a value from r->pool -- in this case connection exists for the lifetime of ptrans, which is longer than r->pool (especially if r->pool is a subrequest!). So the correct thing to do is to allocate from connection->pool.

And there was another interesting bug in [mod\\_include](#) / [mod\\_cgi](#). You'll see in those that they do this test to decide if they should use r->pool or r->main->pool. In this case the resource that they are registering for cleanup is a child process. If

it were registered in `r->pool`, then the code would `wait()` for the child when the subrequest finishes. With `mod_include` this could be any old `#include`, and the delay can be up to 3 seconds... and happened quite frequently. Instead the subprocess is registered in `r->main->pool` which causes it to be cleaned up when the entire request is done -- *i.e.*, after the output has been sent to the client and logging has happened.

## Tracking open files, etc.

As indicated above, resource pools are also used to track other sorts of resources besides memory. The most common are open files. The routine which is typically used for this is `ap_pfdopen`, which takes a resource pool and two strings as arguments; the strings are the same as the typical arguments to `fopen`, *e.g.*,

```
...  
FILE *f = ap_pfdopen (r->pool, r->filename, "r");  
if (f == NULL) { ... } else { ... }
```

There is also a `ap_popenf` routine, which parallels the lower-level open system call. Both of these routines arrange for the file to be closed when the resource pool in question is cleared.

Unlike the case for memory, there *are* functions to close files allocated with `ap_pfdopen`, and `ap_popenf`, namely `ap_pfdclose` and `ap_pclosef`. (This is because, on many systems, the number of files which a single process can have open is quite limited). It is important to use these functions to close files allocated with `ap_pfdopen` and `ap_popenf`, since to do otherwise could cause fatal errors on systems such as Linux, which react badly if the same `FILE*` is closed more than once.

(Using the `close` functions is not mandatory, since the file will

eventually be closed regardless, but you should consider it in cases where your module is opening, or could open, a lot of files).

## **Other sorts of resources -- cleanup functions**

More text goes here. Describe the cleanup primitives in terms of which the file stuff is implemented; also, `spawn_process`.

Pool cleanups live until `clear_pool()` is called:

`clear_pool(a)` recursively calls `destroy_pool()` on all subpools of `a`; then calls all the cleanups for `a`; then releases all the memory for `a`. `destroy_pool(a)` calls `clear_pool(a)` and then releases the pool structure itself. *i.e.*, `clear_pool(a)` doesn't delete `a`, it just frees up all the resources and you can start using it again immediately.

## **Fine control -- creating and dealing with sub-pools, with a note on sub-requests**

On rare occasions, too-free use of `ap_palloc()` and the associated primitives may result in undesirably profligate resource allocation. You can deal with such a case by creating a *sub-pool*, allocating within the sub-pool rather than the main pool, and clearing or destroying the sub-pool, which releases the resources which were associated with it. (This really *is* a rare situation; the only case in which it comes up in the standard module set is in case of listing directories, and then only with *very* large directories. Unnecessary use of the primitives discussed here can hair up your code quite a bit, with very little gain).

The primitive for creating a sub-pool is `ap_make_sub_pool`, which takes another pool (the parent pool) as an argument. When the main pool is cleared, the sub-pool will be destroyed. The sub-pool may also be cleared or destroyed at any time, by calling the functions `ap_clear_pool` and `ap_destroy_pool`, respectively.

(The difference is that `ap_clear_pool` frees resources associated with the pool, while `ap_destroy_pool` also deallocates the pool itself. In the former case, you can allocate new resources within the pool, and clear it again, and so forth; in the latter case, it is simply gone).

One final note -- sub-requests have their own resource pools, which are sub-pools of the resource pool for the main request. The polite way to reclaim the resources associated with a sub request which you have allocated (using the `ap_sub_req_...` functions) is `ap_destroy_sub_req`, which frees the resource pool. Before calling this function, be sure to copy anything that you care about which might be allocated in the sub-request's resource pool into someplace a little less volatile (for instance, the filename in its `request_rec` structure).

(Again, under most circumstances, you shouldn't feel obliged to call this function; only 2K of memory or so are allocated for a typical sub request, and it will be freed anyway when the main request pool is cleared. It is only when you are allocating many, many sub-requests for a single main request that you should seriously consider the `ap_destroy_...` functions).



## Configuration, Commands and the like

One of the design goals for this server was to maintain external compatibility with the NCSA 1.3 server --- that is, to read the same configuration files, to process all the directives therein correctly, and in general to be a drop-in replacement for NCSA. On the other hand, another design goal was to move as much of the server's functionality into modules which have as little as possible to do with the monolithic server core. The only way to reconcile these goals is to move the handling of most commands from the central server into the modules.

However, just giving the modules command tables is not enough to divorce them completely from the server core. The server has to remember the commands in order to act on them later. That involves maintaining data which is private to the modules, and which can be either per-server, or per-directory. Most things are per-directory, including in particular access control and authorization information, but also information on how to determine file types from suffixes, which can be modified by [AddType](#) and [ForceType](#) directives, and so forth. In general, the governing philosophy is that anything which *can* be made configurable by directory should be; per-server information is generally used in the standard set of modules for information like [Aliases](#) and [Redirects](#) which come into play before the request is tied to a particular place in the underlying file system.

Another requirement for emulating the NCSA server is being able to handle the per-directory configuration files, generally called `.htaccess` files, though even in the NCSA server they can contain directives which have nothing at all to do with access control. Accordingly, after URI -> filename translation, but before performing any other phase, the server walks down the directory hierarchy of the underlying filesystem, following the translated pathname, to read any `.htaccess` files which might be present.

The information which is read in then has to be *merged* with the applicable information from the server's own config files (either from the `<Directory>` sections in `access.conf`, or from defaults in `srml.conf`, which actually behaves for most purposes almost exactly like `<Directory />`).

Finally, after having served a request which involved reading `.htaccess` files, we need to discard the storage allocated for handling them. That is solved the same way it is solved wherever else similar problems come up, by tying those structures to the per-transaction resource pool.

## Per-directory configuration structures

Let's look out how all of this plays out in `mod_mime.c`, which defines the file typing handler which emulates the NCSA server's behavior of determining file types from suffixes. What we'll be looking at, here, is the code which implements the `AddType` and `AddEncoding` commands. These commands can appear in `.htaccess` files, so they must be handled in the module's private per-directory data, which in fact, consists of two separate tables for MIME types and encoding information, and is declared as follows:

```
typedef struct {
    table *forced_types;      /* Additional AddTyped stuff */
    table *encoding_types;   /* Added with AddEncoding... */
} mime_dir_config;
```

When the server is reading a configuration file, or `<Directory>` section, which includes one of the MIME module's commands, it needs to create a `mime_dir_config` structure, so those commands have something to act on. It does this by invoking the function it finds in the module's 'create per-dir config slot', with two arguments: the name of the directory to which this configuration

information applies (or NULL for `srm.conf`), and a pointer to a resource pool in which the allocation should happen.

(If we are reading a `.htaccess` file, that resource pool is the per-request resource pool for the request; otherwise it is a resource pool which is used for configuration data, and cleared on restarts. Either way, it is important for the structure being created to vanish when the pool is cleared, by registering a cleanup on the pool if necessary).

For the MIME module, the per-dir config creation function just `ap_pallocs` the structure above, and creates a couple of tables to fill it. That looks like this:

```
void *create_mime_dir_config (pool *p, char *dummy)
{
    mime_dir_config *new =
        (mime_dir_config *) ap_palloc (p,
        sizeof(mime_dir_config));

    new->forced_types = ap_make_table (p, 4);
    new->encoding_types = ap_make_table (p, 4);

    return new;
}
```

Now, suppose we've just read in a `.htaccess` file. We already have the per-directory configuration structure for the next directory up in the hierarchy. If the `.htaccess` file we just read in didn't have any [AddType](#) or [AddEncoding](#) commands, its per-directory config structure for the MIME module is still valid, and we can just use it. Otherwise, we need to merge the two structures somehow.

To do that, the server invokes the module's per-directory config merge function, if one is present. That function takes three arguments: the two structures being merged, and a resource pool in which to allocate the result. For the MIME module, all that needs

to be done is overlay the tables from the new per-directory config structure with those from the parent:

```
void *merge_mime_dir_configs (pool *p, void *parent_dirv, void
*subdirv)
{
    mime_dir_config *parent_dir = (mime_dir_config
*)parent_dirv;
    mime_dir_config *subdir = (mime_dir_config *)subdirv;
    mime_dir_config *new =
        (mime_dir_config *)ap_palloc (p, sizeof(mime_dir_config));

    new->forced_types = ap_overlay_tables (p, subdir-
>forced_types,
        parent_dir->forced_types);
    new->encoding_types = ap_overlay_tables (p, subdir-
>encoding_types,
        parent_dir->encoding_types);

    return new;
}
```

As a note -- if there is no per-directory merge function present, the server will just use the subdirectory's configuration info, and ignore the parent's. For some modules, that works just fine (e.g., for the includes module, whose per-directory configuration information consists solely of the state of the XBITHACK), and for those modules, you can just not declare one, and leave the corresponding structure slot in the module itself NULL.

## Command handling

Now that we have these structures, we need to be able to figure out how to fill them. That involves processing the actual [AddType](#) and [AddEncoding](#) commands. To find commands, the server looks in the module's command table. That table contains information on how many arguments the commands take, and in what formats, where it is permitted, and so forth. That information is sufficient to allow the server to invoke most command-handling functions with pre-parsed arguments. Without further ado, let's

look at the [AddType](#) command handler, which looks like this (the [AddEncoding](#) command looks basically the same, and won't be shown here):

```
char *add_type(cmd_parms *cmd, mime_dir_config *m, char *ct,
char *ext)
{
    if (*ext == '.') ++ext;
    ap_table_set (m->forced_types, ext, ct);
    return NULL;
}
```

This command handler is unusually simple. As you can see, it takes four arguments, two of which are pre-parsed arguments, the third being the per-directory configuration structure for the module in question, and the fourth being a pointer to a `cmd_parms` structure. That structure contains a bunch of arguments which are frequently of use to some, but not all, commands, including a resource pool (from which memory can be allocated, and to which cleanups should be tied), and the (virtual) server being configured, from which the module's per-server configuration data can be obtained if required.

Another way in which this particular command handler is unusually simple is that there are no error conditions which it can encounter. If there were, it could return an error message instead of `NULL`; this causes an error to be printed out on the server's `stderr`, followed by a quick exit, if it is in the main config files; for a `.htaccess` file, the syntax error is logged in the server error log (along with an indication of where it came from), and the request is bounced with a server error response (HTTP error status, code 500).

The MIME module's command table has entries for these commands, which look like this:

```

command_rec mime_cmds[] = {
    { "AddType", add_type, NULL, OR_FILEINFO, TAKE2,
      "a mime type followed by a file extension" },
    { "AddEncoding", add_encoding, NULL, OR_FILEINFO, TAKE2,
      "an encoding (e.g., gzip), followed by a file extension"
    },
    { NULL }
};

```

The entries in these tables are:

- The name of the command
- The function which handles it
- a (`void *`) pointer, which is passed in the `cmd_parms` structure to the command handler --- this is useful in case many similar commands are handled by the same function.
- A bit mask indicating where the command may appear. There are mask bits corresponding to each `AllowOverride` option, and an additional mask bit, `RSRC_CONF`, indicating that the command may appear in the server's own config files, but *not* in any `.htaccess` file.
- A flag indicating how many arguments the command handler wants pre-parsed, and how they should be passed in. `TAKE2` indicates two pre-parsed arguments. Other options are `TAKE1`, which indicates one pre-parsed argument, `FLAG`, which indicates that the argument should be `On` or `Off`, and is passed in as a boolean flag, `RAW_ARGS`, which causes the server to give the command the raw, unparsed arguments (everything but the command name itself). There is also `ITERATE`, which means that the handler looks the same as `TAKE1`, but that if multiple arguments are present, it should be called multiple times, and finally `ITERATE2`, which indicates that the command handler looks like a `TAKE2`, but if more arguments are present, then it should be called multiple times, holding the first argument constant.

- Finally, we have a string which describes the arguments that should be present. If the arguments in the actual config file are not as required, this string will be used to help give a more specific error message. (You can safely leave this NULL).

Finally, having set this all up, we have to use it. This is ultimately done in the module's handlers, specifically for its file-typing handler, which looks more or less like this; note that the per-directory configuration structure is extracted from the request\_rec's per-directory configuration vector by using the ap\_get\_module\_config function.

```
int find_ct(request_rec *r)
{
    int i;
    char *fn = ap_pstrdup (r->pool, r->filename);
    mime_dir_config *conf = (mime_dir_config *)
        ap_get_module_config(r->per_dir_config, &mime_module);
    char *type;

    if (S_ISDIR(r->finfo.st_mode)) {
        r->content_type = DIR_MAGIC_TYPE;
        return OK;
    }

    if((i=ap_rind(fn, '.')) < 0) return DECLINED;
    ++i;

    if ((type = ap_table_get (conf->encoding_types, &fn[i])))
    {
        r->content_encoding = type;

        /* go back to previous extension to try to use it as a
        type */
        fn[i-1] = '\0';
        if((i=ap_rind(fn, '.')) < 0) return OK;
        ++i;
    }

    if ((type = ap_table_get (conf->forced_types, &fn[i])))
    {
        r->content_type = type;
    }
}
```

```
    return OK;
}
```

## Side notes -- per-server configuration, virtual servers, etc.

The basic ideas behind per-server module configuration are basically the same as those for per-directory configuration; there is a creation function and a merge function, the latter being invoked where a virtual server has partially overridden the base server configuration, and a combined structure must be computed. (As with per-directory configuration, the default if no merge function is specified, and a module is configured in some virtual server, is that the base configuration is simply ignored).

The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the `alias` module, which also indicates how a syntax error can be returned (note that the per-directory configuration argument to the command handler is declared as a dummy, since the module doesn't actually have per-directory config data):

```
char *add_redirect(cmd_parms *cmd, void *dummy, char *f, char
*url)
{
    server_rec *s = cmd->server;
    alias_server_conf *conf = (alias_server_conf *)
        ap_get_module_config(s->module_config, &alias_module);
    alias_entry *new = ap_push_array (conf->redirects);

    if (!ap_is_url (url)) return "Redirect to non-URL";

    new->fake = f; new->real = url;
    return NULL;
}
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

# Debugging Memory Allocation in APR

This document has been removed.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

## Documenting code in Apache 2.4

Apache 2.4 uses [Doxygen](#) to document the APIs and global variables in the code. This will explain the basics of how to document using Doxygen.



To start a documentation block, use `/**`

To end a documentation block, use `*/`

In the middle of the block, there are multiple tags we can use:

```
Description of this functions purpose
@param parameter_name description
@return description
@deffunc signature of the function
```

The `deffunc` is not always necessary. Doxygen does not have a full parser in it, so any prototype that use a macro in the return type declaration is too complex for scandoc. Those functions require a `deffunc`. An example (using `&gt;` rather than `>`):

```
/**
 * return the final element of the pathname
 * @param pathname The path to get the final element of
 * @return the final element of the path
 * @tip Examples:
 * <pre>
 * "/foo/bar/gum" -&gt; "gum"
 * "/foo/bar/gum/" -&gt; ""
 * "gum" -&gt; "gum"
 * "wi\\n32\\stuff" -&gt; "stuff"
 * </pre>
 * @deffunc const char * ap_filename_of_pathname(const char
 *pathname)
 */
```

At the top of the header file, always include:

```
/**
 * @package Name of library header
 */
```

Doxygen uses a new HTML file for each package. The HTML files are named `{Name_of_library_header}.html`, so try to be concise with your names.

For a further discussion of the possibilities please refer to [the Doxygen site](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

# Hook Functions in the Apache HTTP Server 2.x

## Warning

This document is still in development and may be partially out of date.

In general, a hook function is one that the Apache HTTP Server will call at some point during the processing of a request. Modules can provide functions that are called, and specify when they get called in comparison to other modules.



---

The httpd's core modules offer a predefined list of hooks used during the standard [request processing](#) phase. Creating a new hook will expose a function that implements it (see sections below) but it is essential to understand that you will not extend the httpd's core hooks. Their presence and order in the request processing is in fact a consequence of how they are called in `server/request.c` (check [this section](#) for an overview). The core hooks are listed in the [doxygen documentation](#).

Reading [guide for developing modules](#) and [request processing](#) before proceeding is highly recommended.



## Creating a hook function

In order to create a new hook, four things need to be done:

### Declare the hook function

Use the `AP_DECLARE_HOOK` macro, which needs to be given the return type of the hook function, the name of the hook, and the arguments. For example, if the hook returns an `int` and takes a `request_rec *` and an `int` and is called `do_something`, then declare it like this:

```
AP_DECLARE_HOOK(int, do_something, (request_rec *, int))
```

This should go in a header which modules will include if they want to use the hook.

### Create the hook structure

Each source file that exports a hook has a private structure which is used to record the module functions that use the hook. This is declared as follows:

```
APR_HOOK_STRUCT(  
    APR_HOOK_LINK(do_something)  
    ...  
)
```

### Implement the hook caller

The source file that exports the hook has to implement a function that will call the hook. There are currently three possible ways to do this. In all cases, the calling function is called `ap_run_hookname()`.

## Void hooks

If the return value of a hook is `void`, then all the hooks are called, and the caller is implemented like this:

```
AP_IMPLEMENT_HOOK_VOID(do_something, (reque
```

The second and third arguments are the dummy argument declaration and the dummy arguments as they will be used when calling the hook. In other words, this macro expands to something like this:

```
void ap_run_do_something(request_rec *r, in
{
    ...
    do_something(r, n);
}
```

## Hooks that return a value

If the hook returns a value, then it can either be run until the first hook that does something interesting, like so:

```
AP_IMPLEMENT_HOOK_RUN_FIRST(int, do_somethi
```

The first hook that does *not* return `DECLINED` stops the loop and its return value is returned from the hook caller. Note that `DECLINED` is the traditional hook return value meaning "I didn't do anything", but it can be whatever suits you.

Alternatively, all hooks can be run until an error occurs. This boils down to permitting *two* return values, one of which means "I did something, and it was OK" and the other meaning "I did nothing".

The first function that returns a value other than one of those two stops the loop, and its return is the return value. Declare these like so:

```
AP_IMPLEMENT_HOOK_RUN_ALL(int, do_something
```

Again, OK and DECLINED are the traditional values. You can use what you want.

## Call the hook callers

At appropriate moments in the code, call the hook caller, like so:

```
int n, ret;  
request_rec *r;  
  
ret=ap_run_do_something(r, n);
```



A module that wants a hook to be called needs to do two things.

### Implement the hook function

Include the appropriate header, and define a static function of the correct type:

```
static int my_something_doer(request_rec *r
{
    ...
    return OK;
}
```

### Add a hook registering function

During initialisation, the server will call each modules hook registering function, which is included in the module structure:

```
static void my_register_hooks()
{
    ap_hook_do_something(my_something_doer,
}

module MODULE_VAR_EXPORT my_module =
{
    ...
    my_register_hooks          /* register hoo
};
```

### Controlling hook calling order

In the example above, we didn't use the three arguments in the hook registration function that control calling order of all the

functions registered within the hook. There are two mechanisms for doing this. The first, rather crude, method, allows us to specify roughly where the hook is run relative to other modules. The final argument control this. There are three possible values:

APR\_HOOK\_FIRST, APR\_HOOK\_MIDDLE and APR\_HOOK\_LAST.

All modules using any particular value may be run in any order relative to each other, but, of course, all modules using APR\_HOOK\_FIRST will be run before APR\_HOOK\_MIDDLE which are before APR\_HOOK\_LAST. Modules that don't care when they are run should use APR\_HOOK\_MIDDLE. *These values are spaced out, so that positions like APR\_HOOK\_FIRST - 2 are possible to hook slightly earlier than other functions.*

Note that there are two more values, APR\_HOOK\_REALLY\_FIRST and APR\_HOOK\_REALLY\_LAST. These should only be used by the hook exporter.

The other method allows finer control. When a module knows that it must be run before (or after) some other modules, it can specify them by name. The second (third) argument is a NULL-terminated array of strings consisting of the names of modules that must be run before (after) the current module. For example, suppose we want "mod\_xyz.c" and "mod\_abc.c" to run before we do, then we'd hook as follows:

```
static void register_hooks()
{
    static const char * const aszPre[] = {
        ap_hook_do_something(my_something_doer,
    }
```

Note that the sort used to achieve this is stable, so ordering set by

`APR_HOOK_ORDER` is preserved, as far as is possible.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

## Converting Modules from Apache 1.3 to Apache 2.0

This is a first attempt at writing the lessons I learned when trying to convert the `mod_mmap_static` module to Apache 2.0. It's by no means definitive and probably won't even be correct in some ways, but it's a start.



## Cleanup Routines

These now need to be of type `apr_status_t` and return a value of that type. Normally the return value will be `APR_SUCCESS` unless there is some need to signal an error in the cleanup. Be aware that even though you signal an error not all code yet checks and acts upon the error.

## Initialisation Routines

These should now be renamed to better signify where they sit in the overall process. So the name gets a small change from `mmap_init` to `mmap_post_config`. The arguments passed have undergone a radical change and now look like

- `apr_pool_t *p`
- `apr_pool_t *plog`
- `apr_pool_t *ptemp`
- `server_rec *s`

## Data Types

A lot of the data types have been moved into the [APR](#). This means that some have had a name change, such as the one shown above. The following is a brief list of some of the changes that you are likely to have to make.

- `pool` becomes `apr_pool_t`
- `table` becomes `apr_table_t`



## Register Hooks

The new architecture uses a series of hooks to provide for calling your functions. These you'll need to add to your module by way of a new function, `static void register_hooks(void)`. The function is really reasonably straightforward once you understand what needs to be done. Each function that needs calling at some stage in the processing of a request needs to be registered, handlers do not. There are a number of phases where functions can be added, and for each you can specify with a high degree of control the relative order that the function will be called in.

This is the code that was added to `mod_mmap_static`:

```
static void register_hooks(void)
{
    static const char * const aszPre[]={ "http_core.c",NULL };
    ap_hook_post_config(mmap_post_config,NULL,NULL,HOOK_MIDDLE);
    ap_hook_translate_name(mmap_static_xlat,aszPre,NULL,HOOK_LAST);
};
```

This registers 2 functions that need to be called, one in the `post_config` stage (virtually every module will need this one) and one for the `translate_name` phase. note that while there are different function names the format of each is identical. So what is the format?

```
ap_hook_phase_name(function_name, predecessors, successors,
position);
```

There are 3 hook positions defined...

- HOOK\_FIRST
- HOOK\_MIDDLE
- HOOK\_LAST

To define the position you use the position and then modify it with the predecessors and successors. Each of the modifiers can be a list of functions that should be called, either before the function is run (predecessors) or after the function has run (successors).

In the `mod_mmap_static` case I didn't care about the `post_config` stage, but the `mmap_static_xlat` **must** be called after the core module had done its name translation, hence the use of the `aszPre` to define a modifier to the position `HOOK_LAST`.

## Module Definition

There are now a lot fewer stages to worry about when creating your module definition. The old definition looked like

```
module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD_MODULE_STUFF,
    /* initializer */
    /* dir config creator */
    /* dir merger --- default is to override */
    /* server config */
    /* merge server config */
    /* command handlers */
    /* handlers */
    /* filename translation */
    /* check_user_id */
    /* check auth */
    /* check access */
    /* type_checker */
    /* fixups */
    /* logger */
    /* header parser */
    /* child_init */
    /* child_exit */
    /* post read-request */
};
```

The new structure is a great deal simpler...

```

module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD20_MODULE_STUFF,
    /* create per-directory config structures */
    /* merge per-directory config structures */
    /* create per-server config structures */
    /* merge per-server config structures */
    /* command handlers */
    /* handlers */
    /* register hooks */
};

```

Some of these read directly across, some don't. I'll try to summarise what should be done below.

The stages that read directly across :

```

/* dir config creator */
    /* create per-directory config structures */
/* server config */
    /* create per-server config structures */
/* dir merger */
    /* merge per-directory config structures */
/* merge server config */
    /* merge per-server config structures */
/* command table */
    /* command apr_table_t */
/* handlers */
    /* handlers */

```

The remainder of the old functions should be registered as hooks. There are the following hook stages defined so far...

### **ap\_hook\_pre\_config**

do any setup required prior to processing configuration directives

**ap\_hook\_check\_config**

review configuration directive interdependencies

**ap\_hook\_test\_config**

executes only with -t option

**ap\_hook\_open\_logs**

open any specified logs

**ap\_hook\_post\_config**

this is where the old `_init` routines get registered

**ap\_hook\_http\_method**

retrieve the http method from a request. (legacy)

**ap\_hook\_auth\_checker**

check if the resource requires authorization

**ap\_hook\_access\_checker**

check for module-specific restrictions

**ap\_hook\_check\_user\_id**

check the user-id and password

**ap\_hook\_default\_port**

retrieve the default port for the server

**ap\_hook\_pre\_connection**

do any setup required just before processing, but after accepting

**ap\_hook\_process\_connection**

run the correct protocol

**ap\_hook\_child\_init**

call as soon as the child is started

**ap\_hook\_create\_request**

??

**ap\_hook\_fixups**

last chance to modify things before generating content

**ap\_hook\_handler**

generate the content

**ap\_hook\_header\_parser**

lets modules look at the headers, not used by most modules, because they use `post_read_request` for this

**ap\_hook\_insert\_filter**

to insert filters into the filter chain

**ap\_hook\_log\_transaction**

log information about the request

**ap\_hook\_optional\_fn\_retrieve**

retrieve any functions registered as optional

**ap\_hook\_post\_read\_request**

called after reading the request, before any other phase

**ap\_hook\_quick\_handler**

called before any request processing, used by cache modules.

**ap\_hook\_translate\_name**

translate the URI into a filename

**ap\_hook\_type\_checker**

determine and/or set the doc type



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

# Request Processing in the Apache HTTP Server 2.x

## Warning

Warning - this is a first (fast) draft that needs further revision!

Several changes in 2.0 and above affect the internal request processing mechanics. Module authors need to be aware of these changes so they may take advantage of the optimizations and security enhancements.

The first major change is to the subrequest and redirect mechanisms. There were a number of different code paths in the Apache HTTP Server 1.3 to attempt to optimize subrequest or redirect behavior. As patches were introduced to 2.0, these optimizations (and the server behavior) were quickly broken due to this duplication of code. All duplicate code has been folded back into `ap_process_request_internal()` to prevent the code from falling out of sync again.

This means that much of the existing code was 'unoptimized'. It is the Apache HTTP Project's first goal to create a robust and correct implementation of the HTTP server RFC. Additional goals include security, scalability and optimization. New methods were sought to optimize the server (beyond the performance of 1.3) without introducing fragile or insecure code.



## The Request Processing Cycle

All requests pass through `ap_process_request_internal()` in `server/request.c`, including subrequests and redirects. If a module doesn't pass generated requests through this code, the author is cautioned that the module may be broken by future changes to request processing.

To streamline requests, the module author can take advantage of the [hooks offered](#) to drop out of the request cycle early, or to bypass core hooks which are irrelevant (and costly in terms of CPU.)



### Unescapes the URL

The request's `parsed_uri` path is unescaped, once and only once, at the beginning of internal request processing.

This step is bypassed if the `proxyreq` flag is set, or the `parsed_uri.path` element is unset. The module has no further control of this one-time unescape operation, either failing to unescape or multiply unescaping the URL leads to security repercussions.

### Strips Parent and This Elements from the URI

All `/../` and `/./` elements are removed by `ap_getparents()`. This helps to ensure the path is (nearly) absolute before the request processing continues.

This step cannot be bypassed.

### Initial URI Location Walk

Every request is subject to an `ap_location_walk()` call. This ensures that `<Location>` sections are consistently enforced for all requests. If the request is an internal redirect or a sub-request, it may borrow some or all of the processing from the previous or parent request's `ap_location_walk`, so this step is generally very efficient after processing the main request.

### `translate_name`

Modules can determine the file name, or alter the given URI in this step. For example, `mod_vhost_alias` will translate the URI's path into the configured virtual host, `mod_alias` will translate the path to an alias path, and if the request falls back on the core, the

[DocumentRoot](#) is prepended to the request resource.

If all modules DECLINE this phase, an error 500 is returned to the browser, and a "couldn't translate name" error is logged automatically.

### **Hook: map\_to\_storage**

After the file or correct URI was determined, the appropriate per-dir configurations are merged together. For example, [mod\\_proxy](#) compares and merges the appropriate [<Proxy>](#) sections. If the URI is nothing more than a local (non-proxy) TRACE request, the core handles the request and returns DONE. If no module answers this hook with OK or DONE, the core will run the request filename against the [<Directory>](#) and [<Files>](#) sections. If the request 'filename' isn't an absolute, legal filename, a note is set for later termination.

### **URI Location Walk**

Every request is hardened by a second `ap_location_walk()` call. This reassures that a translated request is still subjected to the configured [<Location>](#) sections. The request again borrows some or all of the processing from its previous `location_walk` above, so this step is almost always very efficient unless the translated URI mapped to a substantially different path or Virtual Host.

### **Hook: header\_parser**

The main request then parses the client's headers. This prepares the remaining request processing steps to better serve the client's request.



Needs Documentation. Code is:

```
if ((access_status = ap_run_access_checker(r))
    return decl_die(access_status, "check a
}

if ((access_status = ap_run_check_user_id(r))
    return decl_die(access_status, "check u
}

if ((access_status = ap_run_auth_checker(r))
    return decl_die(access_status, "check a
}
```



## Hook: `type_checker`

The modules have an opportunity to test the URI or filename against the target resource, and set mime information for the request. Both `mod_mime` and `mod_mime_magic` use this phase to compare the file name or contents against the administrator's configuration and set the content type, language, character set and request handler. Some modules may set up their filters or other request handling parameters at this time.

If all modules `DECLINE` this phase, an error 500 is returned to the browser, and a "couldn't find types" error is logged automatically.

## Hook: `fixups`

Many modules are 'trounced' by some phase above. The `fixups` phase is used by modules to 'reassert' their ownership or force the request's fields to their appropriate values. It isn't always the cleanest mechanism, but occasionally it's the only option.



## THE HANDLER PHASE

This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

### Hook: `insert_filter`

Modules that transform the content in some way can insert their values and override existing filters, such that if the user configured a more advanced filter out-of-order, then the module can move its order as need be. There is no result code, so actions in this hook better be trusted to always succeed.

### Hook: `handler`

The module finally has a chance to serve the request in its handler hook. Note that not every prepared request is sent to the handler hook. Many modules, such as `mod_autoindex`, will create subrequests for a given URI, and then never serve the subrequest, but simply lists it for the user. Remember not to put required teardown from the hooks above into this module, but register pool cleanups against the request pool to free resources as required.



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

## How filters work in Apache 2.0

### Warning

This is a cut 'n paste job from an email (<022501c1c529\$63a9550\$7f00000a@KOJ>) and only reformatted for better readability. It's not up to date but may be a good start for further research.



## Filter types

There are three basic filter types (each of these is actually broken down into two categories, but that comes later).

### **CONNECTION**

Filters of this type are valid for the lifetime of this connection. (AP\_FTYPE\_CONNECTION, AP\_FTYPE\_NETWORK)

### **PROTOCOL**

Filters of this type are valid for the lifetime of this request from the point of view of the client, this means that the request is valid from the time that the request is sent until the time that the response is received. (AP\_FTYPE\_PROTOCOL, AP\_FTYPE\_TRANSCODE)

### **RESOURCE**

Filters of this type are valid for the time that this content is used to satisfy a request. For simple requests, this is identical to PROTOCOL, but internal redirects and sub-requests can change the content without ending the request. (AP\_FTYPE\_RESOURCE, AP\_FTYPE\_CONTENT\_SET)

It is important to make the distinction between a protocol and a resource filter. A resource filter is tied to a specific resource, it may also be tied to header information, but the main binding is to a resource. If you are writing a filter and you want to know if it is resource or protocol, the correct question to ask is: "Can this filter be removed if the request is redirected to a different resource?" If the answer is yes, then it is a resource filter. If it is no, then it is most likely a protocol or connection filter. I won't go into connection filters, because they seem to be well understood. With this definition, a few examples might help:

### **Byterange**

We have coded it to be inserted for all requests, and it is removed if not used. Because this filter is active at the

beginning of all requests, it can not be removed if it is redirected, so this is a protocol filter.

### **http\_header**

This filter actually writes the headers to the network. This is obviously a required filter (except in the asis case which is special and will be dealt with below) and so it is a protocol filter.

### **Deflate**

The administrator configures this filter based on which file has been requested. If we do an internal redirect from an autoindex page to an index.html page, the deflate filter may be added or removed based on config, so this is a resource filter.

The further breakdown of each category into two more filter types is strictly for ordering. We could remove it, and only allow for one filter type, but the order would tend to be wrong, and we would need to hack things to make it work. Currently, the RESOURCE filters only have one filter type, but that should change.



---

This is actually rather simple in theory, but the code is complex. First of all, it is important that everybody realize that there are three filter lists for each request, but they are all concatenated together:

- `r->output_filters` (corresponds to RESOURCE)
- `r->proto_output_filters` (corresponds to PROTOCOL)
- `r->connection->output_filters` (corresponds to CONNECTION)

The problem previously, was that we used a singly linked list to create the filter stack, and we started from the "correct" location. This means that if I had a RESOURCE filter on the stack, and I added a CONNECTION filter, the CONNECTION filter would be ignored. This should make sense, because we would insert the connection filter at the top of the `c->output_filters` list, but the end of `r->output_filters` pointed to the filter that used to be at the front of `c->output_filters`. This is obviously wrong. The new insertion code uses a doubly linked list. This has the advantage that we never lose a filter that has been inserted. Unfortunately, it comes with a separate set of headaches.

The problem is that we have two different cases where we use subrequests. The first is to insert more data into a response. The second is to replace the existing response with an internal redirect. These are two different cases and need to be treated as such.

In the first case, we are creating the subrequest from within a handler or filter. This means that the next filter should be passed to `make_sub_request` function, and the last resource filter in the sub-request will point to the next filter in the main request. This makes sense, because the sub-request's data needs to flow through the same set of filters as the main request. A graphical

representation might help:

```
Default_handler --> includes_filter --> byterange --> ...
```

If the includes filter creates a sub request, then we don't want the data from that sub-request to go through the includes filter, because it might not be SSI data. So, the subrequest adds the following:

```
Default_handler --> includes_filter -/-> byterange --> ...  
/   
Default_handler --> sub_request_core
```

What happens if the subrequest is SSI data? Well, that's easy, the `includes_filter` is a resource filter, so it will be added to the sub request in between the `Default_handler` and the `sub_request_core` filter.

The second case for sub-requests is when one sub-request is going to become the real request. This happens whenever a sub-request is created outside of a handler or filter, and NULL is passed as the next filter to the `make_sub_request` function.

In this case, the resource filters no longer make sense for the new request, because the resource has changed. So, instead of starting from scratch, we simply point the front of the resource filters for the sub-request to the front of the protocol filters for the old request. This means that we won't lose any of the protocol filters, neither will we try to send this data through a filter that shouldn't see it.

The problem is that we are using a doubly-linked list for our filter stacks now. But, you should notice that it is possible for two lists to intersect in this model. So, you do you handle the previous pointer? This is a very difficult question to answer, because there

is no "right" answer, either method is equally valid. I looked at why we use the previous pointer. The only reason for it is to allow for easier addition of new servers. With that being said, the solution I chose was to make the previous pointer always stay on the original request.

This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.



The final topic. :-) Mod\_Asis is a bit of a hack, but the handler needs to remove all filters except for connection filters, and send the data. If you are using mod\_asis, all other bets are off.



The absolutely last point is that the reason this code was so hard to get right, was because we had hacked so much to force it to work. I wrote most of the hacks originally, so I am very much to blame. However, now that the code is right, I have started to remove some hacks. Most people should have seen that the `reset_filters` and `add_required_filters` functions are gone. Those inserted protocol level filters for error conditions, in fact, both functions did the same thing, one after the other, it was really strange. Because we don't lose protocol filters for error cases any more, those hacks went away. The `HTTP_HEADER`, `Content-length`, and `Byterange` filters are all added in the `insert_filters` phase, because if they were added earlier, we had some interesting interactions. Now, those could all be moved to be inserted with the `HTTP_IN`, `CORE`, and `CORE_IN` filters. That would make the code easier to follow.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



• • • • •  
, , , , , ( ; , , , , .)



---

## (Access Control)

. URL .  
: [, ,](#)

## (Algorithm)

. (Ciphers)  
.

## APache eXtension Tool (apxs)

[\(module\)](#) ( [DSO](#)) perl .  
: [Manpage: apxs](#)

## (Authentication)

, , .  
: [, ,](#)

## (Certificate)

. (subject ),  
[\(Certificate Authority\)](#) (issuer ), , CA  
X.509 . CA .  
: [SSL/TLS](#)

## (Certificate Signing Request , CSR)

[\(Certification Authority\)](#) CA (Certificate) [\(Private](#)  
[Key\)](#) . CSR .  
: [SSL/TLS](#)

## (Certification Authority , CA)

. CA  
.  
: [SSL/TLS](#)

## (Cipher)

. , DES, IDEA, RC4 .  
: [SSL/TLS](#)

## (Ciphertext)

[\(Plaintext\)](#) [\(Cipher\)](#) .

: [SSL/TLS](#)

(Common Gateway Interface , CGI)

[NCSA](#)

, [RFC](#) .

: [CGI](#)

(Configuration Directive)

:

(Configuration File)

[\(directive\)](#) .

:

**CONNECT**

HTTP HTTP [\(method\)](#). SSL

.

(Context)

[\(configuration file\)](#) [\(directive\)](#) .

:

(Digital Signature)

. [\(Certification Authority\)](#)

(Certificate)

(Public Key)

(Private Ke

. CA , CA

: [SSL/TLS](#)

(Directive)

. [\(Configuration File\)](#) .

:

(Dynamic Shared Object) (DSO)

httpd

[\(Mod](#)

:

(Environment Variable) (env-variable)

. ,

.

:           

**(Export-Crippled)**

(Export Administration Regulations, EAR)

( )  
(Ciphertex

force) .

: SSL/TLS (SSL/TLS Encryption)

**(Filter)**

. , INCLUDES Server Side Includes

:  
:

**(Fully-Qualified Domain-Name) (FQDN)**

IP , . , www  
example.com , www.example.com .

**(Handler)**

. , "(handled)".  
, cgi-script CGI .  
:           

**(Header)**

HTTP .

**.htaccess**

(configuration file) , (directive)

:  
:

**httpd.conf**

(configuration file) .

/usr/local/apache2/conf/httpd.conf,

:  
:

**HyperText Transfer Protocol (HTTP)**

[RFC 2616](#) HTTP/1.1 1.1

## HTTPS

, HyperText Transfer Protocol (Sec  
[SSL](#) HTTP.  
: [SSL/TLS](#)

### (Method)

[HTTP](#) . HTTP GET, POST, PUT

### (Message Digest)

: [SSL/TLS](#)

## MIME-type

. Multipurpose Internet Mail Extensions  
. major type minor type . ,  
text/html, image/gif, application/octet-stream  
. MIME-type HTTP Content-Type [\(header\)](#) .  
: [mod\\_mime](#)

### (Module)

. httpd  
, [DSO](#)  
. base . [\(tarball\)](#)  
(third-party)  
:

### (Module Magic Number) (MMN)

,  
, API . MMN

## OpenSSL

SSL/TLS  
<http://www.openssl.org/>

## Pass Phrase

.  
(Ciphers) / .

: [SSL/TLS](#)

## (Plaintext)

.

## (Private Key)

[\(Public Key Cryptography\)](#) .

: [SSL/TLS](#)

## (Proxy)

. ,

.

.

: [mod\\_proxy](#)

## (Public Key)

[\(Public Key Cryptography\)](#)

.

: [SSL/TLS](#)

## (Public Key Cryptography)

(asymmetric) .

(key pair) . .

: [SSL/TLS](#)

## (Regular Expression) (Regex)

. , " A " , " 10 " ,

" Q " .

. , "images"

.gif.jpg " /images/.\*(jpg|gif)\$" .

[PCRE](#) Perl .

## (Reverse Proxy)

[\(proxy\)](#) .

.

## Secure Sockets Layer (SSL)

Netscape Communications TCP/IP

. *HTTPS* (HyperText Transfer Protocol) over SSL).

: [SSL/TLS](#)

## Server Side Includes (SSI)

HTML .

: [Server Side Includes](#)

## (Session)

(context) .

## SSLeay

Eric A. Young SSL/TLS

## (Symmetric Cryptography)

(*Ciphers*) .

: [SSL/TLS Encryption](#)

## (Tarball)

tar . tar pkzip .

## Transport Layer Security (TLS)

(Internet Engineering Task Force, IETF) TCP/IP

SSL . TLS 1 SSL 3

.  
: [SSL/TLS](#)

## Uniform Resource Locator (URL)

/. [Uniform Resource Identifier](#)

. URL http https (scheme), , .

URL

<http://httpd.apache.org/docs/2.4/glossary.html>

.

## Uniform Resource Identifier (URI)

. [RFC 2396](#) . URI

[URL](#) .

## (Virtual Hosting)

. IP IP . (name-based)  
IP .  
:

### X.509

(International Telecommunication Union, ITU-T) .  
SSL/TLS .  
: [SSL/TLS](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >

\_ .

A | B | C | D | E | F | G | H | I | K | L | M | N | O | P |  
Q | R | S | T | U | V | W | X

- [AcceptFilter](#)
- [AcceptPathInfo](#)
- [AccessFileName](#)
- [Action](#)
- [AddAlt](#)
- [AddAltByEncoding](#)
- [AddAltByType](#)
- [AddCharset](#)
- [AddDefaultCharset](#)
- [AddDescription](#)
- [AddEncoding](#)
- [AddHandler](#)
- [AddIcon](#)
- [AddIconByEncoding](#)
- [AddIconByType](#)
- [AddInputFilter](#)
- [AddLanguage](#)
- [AddModuleInfo](#)
- [AddOutputFilter](#)
- [AddOutputFilterByType](#)
- [AddType](#)
- [Alias](#)
- [AliasMatch](#)
- [Allow](#)
- [AllowCONNECT](#)

- [AllowEncodedSlashes](#)
- [AllowMethods](#)
- [AllowOverride](#)
- [AllowOverrideList](#)
- [Anonymous](#)
- [Anonymous\\_LogEmail](#)
- [Anonymous\\_MustGiveEmail](#)
- [Anonymous\\_NoUserID](#)
- [Anonymous\\_VerifyEmail](#)
- [AsyncRequestWorkerFactor](#)
- [AuthBasicAuthoritative](#)
- [AuthBasicFake](#)
- [AuthBasicProvider](#)
- [AuthBasicUseDigestAlgorithm](#)
- [AuthDBDUserPWQuery](#)
- [AuthDBDUserRealmQuery](#)
- [AuthDBMGroupFile](#)
- [AuthDBMType](#)
- [AuthDBMUserFile](#)
- [AuthDigestAlgorithm](#)
- [AuthDigestDomain](#)
- [AuthDigestNonceLifetime](#)
- [AuthDigestProvider](#)
- [AuthDigestQop](#)
- [AuthDigestShmemSize](#)
- [AuthFormAuthoritative](#)
- [AuthFormBody](#)
- [AuthFormDisableNoStore](#)
- [AuthFormFakeBasicAuth](#)
- [AuthFormLocation](#)
- [AuthFormLoginRequiredLocation](#)
- [AuthFormLoginSuccessLocation](#)
- [AuthFormLogoutLocation](#)
- [AuthFormMethod](#)

- [AuthFormMimetype](#)
- [AuthFormPassword](#)
- [AuthFormProvider](#)
- [AuthFormSitePassphrase](#)
- [AuthFormSize](#)
- [AuthFormUsername](#)
- [AuthGroupFile](#)
- [AuthLDAPAuthorizePrefix](#)
- [AuthLDAPBindAuthoritative](#)
- [AuthLDAPBindDN](#)
- [AuthLDAPBindPassword](#)
- [AuthLDAPCharsetConfig](#)
- [AuthLDAPCompareAsUser](#)
- [AuthLDAPCompareDNOnServer](#)
- [AuthLDAPDereferenceAliases](#)
- [AuthLDAPGroupAttribute](#)
- [AuthLDAPGroupAttributeIsDN](#)
- [AuthLDAPInitialBindAsUser](#)
- [AuthLDAPInitialBindPattern](#)
- [AuthLDAPMaxSubGroupDepth](#)
- [AuthLDAPRemoteUserAttribute](#)
- [AuthLDAPRemoteUserIsDN](#)
- [AuthLDAPSearchAsUser](#)
- [AuthLDAPSubGroupAttribute](#)
- [AuthLDAPSubGroupClass](#)
- [AuthLDAPUrl](#)
- [AuthMerging](#)
- [AuthName](#)
- [AuthnCacheContext](#)
- [AuthnCacheEnable](#)
- [AuthnCacheProvideFor](#)
- [AuthnCacheSOCache](#)
- [AuthnCacheTimeout](#)
- [<AuthnProviderAlias>](#)

- [AuthnzFcgiCheckAuthnProvider](#)
- [AuthnzFcgiDefineProvider](#)
- [AuthType](#)
- [AuthUserFile](#)
- [AuthzDBDLoginToReferer](#)
- [AuthzDBDQuery](#)
- [AuthzDBDRedirectQuery](#)
- [AuthzDBMType](#)
- [<AuthzProviderAlias>](#)
- [AuthzSendForbiddenOnFailure](#)
- [BalancerGrowth](#)
- [BalancerInherit](#)
- [BalancerMember](#)
- [BalancerPersist](#)
- [BrotliAlterETag](#)
- [BrotliCompressionMaxInputBlock](#)
- [BrotliCompressionQuality](#)
- [BrotliCompressionWindow](#)
- [BrotliFilterNote](#)
- [BrowserMatch](#)
- [BrowserMatchNoCase](#)
- [BufferedLogs](#)
- [BufferSize](#)
- [CacheDefaultExpire](#)
- [CacheDetailHeader](#)
- [CacheDirLength](#)
- [CacheDirLevels](#)
- [CacheDisable](#)
- [CacheEnable](#)
- [CacheFile](#)
- [CacheHeader](#)
- [CacheIgnoreCacheControl](#)
- [CacheIgnoreHeaders](#)
- [CacheIgnoreNoLastMod](#)

- [CacheIgnoreQueryString](#)
- [CacheIgnoreURLSessionIdentifiers](#)
- [CacheKeyBaseURL](#)
- [CacheLastModifiedFactor](#)
- [CacheLock](#)
- [CacheLockMaxAge](#)
- [CacheLockPath](#)
- [CacheMaxExpire](#)
- [CacheMaxFileSize](#)
- [CacheMinExpire](#)
- [CacheMinFileSize](#)
- [CacheNegotiatedDocs](#)
- [CacheQuickHandler](#)
- [CacheReadSize](#)
- [CacheReadTime](#)
- [CacheRoot](#)
- [CacheSocache](#)
- [CacheSocacheMaxSize](#)
- [CacheSocacheMaxTime](#)
- [CacheSocacheMinTime](#)
- [CacheSocacheReadSize](#)
- [CacheSocacheReadTime](#)
- [CacheStaleOnError](#)
- [CacheStoreExpired](#)
- [CacheStoreNoStore](#)
- [CacheStorePrivate](#)
- [CGIDScriptTimeout](#)
- [CGIMapExtension](#)
- [CGIPassAuth](#)
- [CGIVar](#)
- [CharsetDefault](#)
- [CharsetOptions](#)
- [CharsetSourceEnc](#)
- [CheckCaseOnly](#)

- [CheckSpelling](#)
- [ChrootDir](#)
- [ContentDigest](#)
- [CookieDomain](#)
- [CookieExpires](#)
- [CookieName](#)
- [CookieStyle](#)
- [CookieTracking](#)
- [CoreDumpDirectory](#)
- [CustomLog](#)
- [Dav](#)
- [DavDepthInfinity](#)
- [DavGenericLockDB](#)
- [DavLockDB](#)
- [DavMinTimeout](#)
- [DBDExptime](#)
- [DBDInitSQL](#)
- [DBDKeep](#)
- [DBDMax](#)
- [DBDMin](#)
- [DBDParams](#)
- [DBDPersist](#)
- [DBDPrepareSQL](#)
- [DBDriver](#)
- [DefaultIcon](#)
- [DefaultLanguage](#)
- [DefaultRuntimeDir](#)
- [DefaultType](#)
- [Define](#)
- [DeflateBufferSize](#)
- [DeflateCompressionLevel](#)
- [DeflateFilterNote](#)
- [DeflateInflateLimitRequestBody](#)
- [DeflateInflateRatioBurst](#)

- [DeflateInflateRatioLimit](#)
- [DeflateMemLevel](#)
- [DeflateWindowSize](#)
- [Deny](#)
- [<Directory>](#)
- [DirectoryCheckHandler](#)
- [DirectoryIndex](#)
- [DirectoryIndexRedirect](#)
- [<DirectoryMatch>](#)
- [DirectorySlash](#)
- [DocumentRoot](#)
- [DTracePrivileges](#)
- [DumpIOInput](#)
- [DumpIOOutput](#)
- [<Else>](#)
- [<Elself>](#)
- [EnableExceptionHook](#)
- [EnableMMAP](#)
- [EnableSendfile](#)
- [Error](#)
- [ErrorDocument](#)
- [ErrorLog](#)
- [ErrorLogFormat](#)
- [Example](#)
- [ExpiresActive](#)
- [ExpiresByType](#)
- [ExpiresDefault](#)
- [ExtendedStatus](#)
- [ExtFilterDefine](#)
- [ExtFilterOptions](#)
- [FallbackResource](#)
- [FileETag](#)
- [<Files>](#)
- [<FilesMatch>](#)

- [FilterChain](#)
- [FilterDeclare](#)
- [FilterProtocol](#)
- [FilterProvider](#)
- [FilterTrace](#)
- [ForceLanguagePriority](#)
- [ForceType](#)
- [ForensicLog](#)
- [GlobalLog](#)
- [GprofDir](#)
- [GracefulShutdownTimeout](#)
- [Group](#)
- [H2CopyFiles](#)
- [H2Direct](#)
- [H2EarlyHints](#)
- [H2MaxSessionStreams](#)
- [H2MaxWorkerIdleSeconds](#)
- [H2MaxWorkers](#)
- [H2MinWorkers](#)
- [H2ModernTLSOnly](#)
- [H2Push](#)
- [H2PushDiarySize](#)
- [H2PushPriority](#)
- [H2PushResource](#)
- [H2SerializeHeaders](#)
- [H2StreamMaxMemSize](#)
- [H2TLSCoolDownSecs](#)
- [H2TLSSwarmUpSize](#)
- [H2Upgrade](#)
- [H2WindowSize](#)
- [Header](#)
- [HeaderName](#)
- [HeartbeatAddress](#)
- [HeartbeatListen](#)

- [HeartbeatMaxServers](#)
- [HeartbeatStorage](#)
- [HeartbeatStorage](#)
- [HostnameLookups](#)
- [HttpProtocolOptions](#)
- [IdentityCheck](#)
- [IdentityCheckTimeout](#)
- [<If>](#)
- [<IfDefine>](#)
- [<IfModule>](#)
- [<IfVersion>](#)
- [ImapBase](#)
- [ImapDefault](#)
- [ImapMenu](#)
- [Include](#)
- [IncludeOptional](#)
- [IndexHeadInsert](#)
- [IndexIgnore](#)
- [IndexIgnoreReset](#)
- [IndexOptions](#)
- [IndexOrderDefault](#)
- [IndexStyleSheet](#)
- [InputSed](#)
- [ISAPIAppendLogToErrors](#)
- [ISAPIAppendLogToQuery](#)
- [ISAPICacheFile](#)
- [ISAPIFakeAsync](#)
- [ISAPILogNotSupported](#)
- [ISAPIReadAheadBuffer](#)
- [KeepAlive](#)
- [KeepAliveTimeout](#)
- [KeptBodySize](#)
- [LanguagePriority](#)
- [LDAPCacheEntries](#)

- [LDAPCacheTTL](#)
- [LDAPConnectionPoolTTL](#)
- [LDAPConnectionTimeout](#)
- [LDAPLibraryDebug](#)
- [LDAPOpCacheEntries](#)
- [LDAPOpCacheTTL](#)
- [LDAPReferralHopLimit](#)
- [LDAPReferrals](#)
- [LDAPRetries](#)
- [LDAPRetryDelay](#)
- [LDAPSharedCacheFile](#)
- [LDAPSharedCacheSize](#)
- [LDAPTimeout](#)
- [LDAPTrustedClientCert](#)
- [LDAPTrustedGlobalCert](#)
- [LDAPTrustedMode](#)
- [LDAPVerifyServerCert](#)
- [<Limit>](#)
- [<LimitExcept>](#)
- [LimitInternalRecursion](#)
- [LimitRequestBody](#)
- [LimitRequestFields](#)
- [LimitRequestFieldSize](#)
- [LimitRequestLine](#)
- [LimitXMLRequestBody](#)
- [Listen](#)
- [ListenBackLog](#)
- [ListenCoresBucketsRatio](#)
- [LoadFile](#)
- [LoadModule](#)
- [<Location>](#)
- [<LocationMatch>](#)
- [LogFormat](#)
- [LogIOTrackTTFB](#)

- [LogLevel](#)
- [LogMessage](#)
- [LuaAuthzProvider](#)
- [LuaCodeCache](#)
- [LuaHookAccessChecker](#)
- [LuaHookAuthChecker](#)
- [LuaHookCheckUserID](#)
- [LuaHookFixups](#)
- [LuaHookInsertFilter](#)
- [LuaHookLog](#)
- [LuaHookMapToStorage](#)
- [LuaHookTranslateName](#)
- [LuaHookTypeChecker](#)
- [LuaInherit](#)
- [LuaInputFilter](#)
- [LuaMapHandler](#)
- [LuaOutputFilter](#)
- [LuaPackageCPath](#)
- [LuaPackagePath](#)
- [LuaQuickHandler](#)
- [LuaRoot](#)
- [LuaScope](#)
- [<Macro>](#)
- [MaxConnectionsPerChild](#)
- [MaxKeepAliveRequests](#)
- [MaxMemFree](#)
- [MaxRangeOverlaps](#)
- [MaxRangeReversals](#)
- [MaxRanges](#)
- [MaxRequestWorkers](#)
- [MaxSpareServers](#)
- [MaxSpareThreads](#)
- [MaxThreads](#)
- [MemcacheConnTTL](#)

- [MergeTrailers](#)
- [MetaDir](#)
- [MetaFiles](#)
- [MetaSuffix](#)
- [MimeMagicFile](#)
- [MinSpareServers](#)
- [MinSpareThreads](#)
- [MMapFile](#)
- [ModemStandard](#)
- [ModMimeUsePathInfo](#)
- [MultiviewsMatch](#)
- [Mutex](#)
- [NameVirtualHost](#)
- [NoProxy](#)
- [NWSSLTrustedCerts](#)
- [NWSSLUpgradeable](#)
- [Options](#)
- [Order](#)
- [OutputSed](#)
- [PassEnv](#)
- [PidFile](#)
- [PrivilegesMode](#)
- [Protocol](#)
- [ProtocolEcho](#)
- [Protocols](#)
- [ProtocolsHonorOrder](#)
- [<Proxy>](#)
- [ProxyAddHeaders](#)
- [ProxyBadHeader](#)
- [ProxyBlock](#)
- [ProxyDomain](#)
- [ProxyErrorOverride](#)
- [ProxyExpressDBMFile](#)
- [ProxyExpressDBMType](#)

- [ProxyExpressEnable](#)
- [ProxyFCGIBackendType](#)
- [ProxyFCGISetEnvIf](#)
- [ProxyFtpDirCharset](#)
- [ProxyFtpEscapeWildcards](#)
- [ProxyFtpListOnWildcard](#)
- [ProxyHCExpr](#)
- [ProxyHCTemplate](#)
- [ProxyHCTPsize](#)
- [ProxyHTMLBufSize](#)
- [ProxyHTMLCharsetOut](#)
- [ProxyHTMLDocType](#)
- [ProxyHTMLEnable](#)
- [ProxyHTMLEvents](#)
- [ProxyHTMLExtended](#)
- [ProxyHTMLFixups](#)
- [ProxyHTMLInterp](#)
- [ProxyHTMLLinks](#)
- [ProxyHTMLMeta](#)
- [ProxyHTMLStripComments](#)
- [ProxyHTMLURLMap](#)
- [ProxyIOBufferSize](#)
- [<ProxyMatch>](#)
- [ProxyMaxForwards](#)
- [ProxyPass](#)
- [ProxyPassInherit](#)
- [ProxyPassInterpolateEnv](#)
- [ProxyPassMatch](#)
- [ProxyPassReverse](#)
- [ProxyPassReverseCookieDomain](#)
- [ProxyPassReverseCookiePath](#)
- [ProxyPreserveHost](#)
- [ProxyReceiveBufferSize](#)
- [ProxyRemote](#)

- [ProxyRemoteMatch](#)
- [ProxyRequests](#)
- [ProxySCGIInternalRedirect](#)
- [ProxySCGISendfile](#)
- [ProxySet](#)
- [ProxySourceAddress](#)
- [ProxyStatus](#)
- [ProxyTimeout](#)
- [ProxyVia](#)
- [QualifyRedirectURL](#)
- [ReadmeName](#)
- [ReceiveBufferSize](#)
- [Redirect](#)
- [RedirectMatch](#)
- [RedirectPermanent](#)
- [RedirectTemp](#)
- [ReflectorHeader](#)
- [RegisterHttpMethod](#)
- [RemoteIPHeader](#)
- [RemoteIPInternalProxy](#)
- [RemoteIPInternalProxyList](#)
- [RemoteIPProxiesHeader](#)
- [RemoteIPTrustedProxy](#)
- [RemoteIPTrustedProxyList](#)
- [RemoveCharset](#)
- [RemoveEncoding](#)
- [RemoveHandler](#)
- [RemoveInputFilter](#)
- [RemoveLanguage](#)
- [RemoveOutputFilter](#)
- [RemoveType](#)
- [RequestHeader](#)
- [RequestReadTimeout](#)
- [Require](#)

- [<RequireAll>](#)
- [<RequireAny>](#)
- [<RequireNone>](#)
- [RewriteBase](#)
- [RewriteCond](#)
- [RewriteEngine](#)
- [RewriteMap](#)
- [RewriteOptions](#)
- [RewriteRule](#)
- [RLimitCPU](#)
- [RLimitMEM](#)
- [RLimitNPROC](#)
- [Satisfy](#)
- [ScoreBoardFile](#)
- [Script](#)
- [ScriptAlias](#)
- [ScriptAliasMatch](#)
- [ScriptInterpreterSource](#)
- [ScriptLog](#)
- [ScriptLogBuffer](#)
- [ScriptLogLength](#)
- [ScriptSock](#)
- [SecureListen](#)
- [SeeRequestTail](#)
- [SendBufferSize](#)
- [ServerAdmin](#)
- [ServerAlias](#)
- [ServerLimit](#)
- [ServerName](#)
- [ServerPath](#)
- [ServerRoot](#)
- [ServerSignature](#)
- [ServerTokens](#)
- [Session](#)

- [SessionCookieName](#)
- [SessionCookieName2](#)
- [SessionCookieRemove](#)
- [SessionCryptoCipher](#)
- [SessionCryptoDriver](#)
- [SessionCryptoPassphrase](#)
- [SessionCryptoPassphraseFile](#)
- [SessionDBDCookieName](#)
- [SessionDBDCookieName2](#)
- [SessionDBDCookieRemove](#)
- [SessionDBDDeleteLabel](#)
- [SessionDBDInsertLabel](#)
- [SessionDBDPerUser](#)
- [SessionDBDSelectLabel](#)
- [SessionDBDUpdateLabel](#)
- [SessionEnv](#)
- [SessionExclude](#)
- [SessionHeader](#)
- [SessionInclude](#)
- [SessionMaxAge](#)
- [SetEnv](#)
- [SetEnvIf](#)
- [SetEnvIfExpr](#)
- [SetEnvIfNoCase](#)
- [SetHandler](#)
- [SetInputFilter](#)
- [SetOutputFilter](#)
- [SSIEndTag](#)
- [SSIErrorMsg](#)
- [SSIETag](#)
- [SSILastModified](#)
- [SSILegacyExprParser](#)
- [SSIStartTag](#)
- [SSITimeFormat](#)

- [SSIUndefinedEcho](#)
- [SSLCACertificateFile](#)
- [SSLCACertificatePath](#)
- [SSLCADNRequestFile](#)
- [SSLCADNRequestPath](#)
- [SSLCARevocationCheck](#)
- [SSLCARevocationFile](#)
- [SSLCARevocationPath](#)
- [SSLCertificateChainFile](#)
- [SSLCertificateFile](#)
- [SSLCertificateKeyFile](#)
- [SSLCipherSuite](#)
- [SSLCompression](#)
- [SSLCryptoDevice](#)
- [SSLEngine](#)
- [SSLFIPS](#)
- [SSLHonorCipherOrder](#)
- [SSLInsecureRenegotiation](#)
- [SSLOCSPDefaultResponder](#)
- [SSLOCSPEnable](#)
- [SSLOCSPNoverify](#)
- [SSLOCSPOverrideResponder](#)
- [SSLOCSPProxyURL](#)
- [SSLOCSPResponderCertificateFile](#)
- [SSLOCSPResponderTimeout](#)
- [SSLOCSPResponseMaxAge](#)
- [SSLOCSPResponseTimeSkew](#)
- [SSLOCSPUseRequestNonce](#)
- [SSLOpenSSLConfCmd](#)
- [SSLOptions](#)
- [SSLPassPhraseDialog](#)
- [SSLProtocol](#)
- [SSLProxyCACertificateFile](#)
- [SSLProxyCACertificatePath](#)

- [SSLProxyCARevocationCheck](#)
- [SSLProxyCARevocationFile](#)
- [SSLProxyCARevocationPath](#)
- [SSLProxyCheckPeerCN](#)
- [SSLProxyCheckPeerExpire](#)
- [SSLProxyCheckPeerName](#)
- [SSLProxyCipherSuite](#)
- [SSLProxyEngine](#)
- [SSLProxyMachineCertificateChainFile](#)
- [SSLProxyMachineCertificateFile](#)
- [SSLProxyMachineCertificatePath](#)
- [SSLProxyProtocol](#)
- [SSLProxyVerify](#)
- [SSLProxyVerifyDepth](#)
- [SSLRandomSeed](#)
- [SSLRenegBufferSize](#)
- [SSLRequire](#)
- [SSLRequireSSL](#)
- [SSLSessionCache](#)
- [SSLSessionCacheTimeout](#)
- [SSLSessionTicketKeyFile](#)
- [SSLSessionTickets](#)
- [SSLSRPUnknownUserSeed](#)
- [SSLSRPVerifierFile](#)
- [SSLStaplingCache](#)
- [SSLStaplingErrorCacheTimeout](#)
- [SSLStaplingFakeTryLater](#)
- [SSLStaplingForceURL](#)
- [SSLStaplingResponderTimeout](#)
- [SSLStaplingResponseMaxAge](#)
- [SSLStaplingResponseTimeSkew](#)
- [SSLStaplingReturnResponderErrors](#)
- [SSLStaplingStandardCacheTimeout](#)
- [SSLStrictSNIVHostCheck](#)

- [SSLUserName](#)
- [SSLUseStapling](#)
- [SSLVerifyClient](#)
- [SSLVerifyDepth](#)
- [StartServers](#)
- [StartThreads](#)
- [Substitute](#)
- [SubstituteInheritBefore](#)
- [SubstituteMaxLineLength](#)
- [Suexec](#)
- [SuexecUserGroup](#)
- [ThreadLimit](#)
- [ThreadsPerChild](#)
- [ThreadStackSize](#)
- [TimeOut](#)
- [TraceEnable](#)
- [TransferLog](#)
- [TypesConfig](#)
- [UnDefine](#)
- [UndefMacro](#)
- [UnsetEnv](#)
- [Use](#)
- [UseCanonicalName](#)
- [UseCanonicalPhysicalPort](#)
- [User](#)
- [UserDir](#)
- [VHostCGIMode](#)
- [VHostCGIPrivs](#)
- [VHostGroup](#)
- [VHostPrivs](#)
- [VHostSecure](#)
- [VHostUser](#)
- [VirtualDocumentRoot](#)
- [VirtualDocumentRootIP](#)

- [<VirtualHost>](#)
- [VirtualScriptAlias](#)
- [VirtualScriptAliasIP](#)
- [WatchdogInterval](#)
- [XBitHack](#)
- [xml2EncAlias](#)
- [xml2EncDefault](#)
- [xml2StartParse](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) >



... ▬

<b>A</b>   <b>B</b>   <b>C</b>   <b>D</b>   <b>E</b>   <b>F</b>   <b>G</b>   <b>H</b>	<b>s</b> <b>v</b> <b>d</b> directory <b>h</b> .htaccess	<b>C</b> Core
<b>I</b>   <b>K</b>   <b>L</b>   <b>M</b>   <b>N</b>   <b>O</b>   <b>P</b>   <b>Q</b>		<b>M</b> MPM
<b>R</b>   <b>S</b>   <b>T</b>   <b>U</b>   <b>V</b>   <b>W</b>   <b>X</b>		<b>B</b> Base <b>E</b> Extension <b>X</b> Experimental <b>T</b> External

<a href="#">AcceptFilter <i>protocol</i> <i>accept</i> <i>filter</i></a>	
Configures optimizations for a Protocol's Listener Sockets	
<a href="#">AcceptPathInfo On Off Default</a>	Default
Resources accept trailing pathname information	
<a href="#">AccessFileName <i>filename</i> [<i>filename</i>] ...</a>	.htaccess
Name of the distributed configuration file	
<a href="#">Action <i>action-type</i> <i>cgi-script</i> [<i>virtual</i>]</a>	
content-type CGI	
<a href="#">AddAlt <i>string</i> <i>file</i> [<i>file</i>] ...</a>	
<a href="#">AddAltByEncoding <i>string</i> <i>MIME-encoding</i> [<i>MIME-encoding</i>] ...</a>	
MIME-encoding	
<a href="#">AddAltByType <i>string</i> <i>MIME-type</i> [<i>MIME-type</i>]</a>	
...	
MIME content-type	

<a href="#">AddCharset <i>charset extension [extension] ...</i></a>	
Maps the given filename extensions to the specified content charset	
<a href="#">AddDefaultCharset On Off <i>charset</i></a>	Off
Default charset parameter to be added when a response content-type is text/plain or text/html	
<a href="#">AddDescription <i>string file [file] ...</i></a>	
<a href="#">AddEncoding <i>encoding extension [extension] ...</i></a>	
Maps the given filename extensions to the specified encoding type	
<a href="#">AddHandler <i>handler-name extension [extension] ...</i></a>	
Maps the filename extensions to the specified handler	
<a href="#">AddIcon <i>icon name [name] ...</i></a>	
<a href="#">AddIconByEncoding <i>icon MIME-encoding [MIME-encoding] ...</i></a>	
MIME content-encoding	
<a href="#">AddIconByType <i>icon MIME-type [MIME-type] ...</i></a>	
MIME content-type	
<a href="#">AddInputFilter <i>filter[:filter...] extension [extension] ...</i></a>	
Maps filename extensions to the filters that will process client requests	
<a href="#">AddLanguage <i>language-tag extension [extension] ...</i></a>	
Maps the given filename extension to the specified content language	
<a href="#">AddModuleInfo <i>module-name string</i></a>	
server-info	
<a href="#">AddOutputFilter <i>filter[:filter...] extension [extension] ...</i></a>	
Maps filename extensions to the filters that will process responses from the server	
<a href="#">AddOutputFilterByType <i>filter[:filter...] media-type [media-type] ...</i></a>	
assigns an output filter to a particular media-type	
<a href="#">AddType <i>media-type extension [extension] ...</i></a>	

Maps the given filename extensions onto the specified content type	
<a href="#">Alias</a> <i>URL-path file-path directory-path</i>	
URL	
<a href="#">AliasMatch</a> <i>regex file-path directory-path</i>	
URL	
<a href="#">Allow from all</a>   <i>host env=[!]<i>env-variable</i></i> <a href="#">[host env=[!]<i>env-variable</i>] ...</a>	
Controls which hosts can access an area of the server	
<a href="#">AllowCONNECT</a> <i>port[-port] [port[-port]] ...</i>	443 563
Ports that are allowed to CONNECT through the proxy	
<a href="#">AllowEncodedSlashes</a> On Off NoDecode	Off
Determines whether encoded path separators in URLs are allowed to be passed through	
<a href="#">AllowMethods</a> <i>reset HTTP-method [HTTP-method]...</i>	reset
Restrict access to the listed HTTP methods	
<a href="#">AllowOverride</a> All None  <i>directive-type</i> <a href="#">[directive-type] ...</a>	None (2.3.9 and lat +
Types of directives that are allowed in .htaccess files	
<a href="#">AllowOverrideList</a> None  <i>directive [directive-type] ...</i>	None
Individual directives that are allowed in .htaccess files	
<a href="#">Anonymous</a> <i>user [user] ...</i>	
<a href="#">Anonymous_ LogEmail</a> On Off	On
<a href="#">Anonymous_ MustGiveEmail</a> On Off	On
<a href="#">Anonymous_ NoUserID</a> On Off	Off
<a href="#">Anonymous_ VerifyEmail</a> On Off	Off
<a href="#">AsyncRequestWorkerFactor</a> <i>factor</i>	
Limit concurrent connections per process	
<a href="#">AuthBasicAuthoritative</a> On Off	On

<a href="#">AuthBasicFake</a> <code>off username [password]</code>	
Fake basic authentication using the given expressions for username and password	
<a href="#">AuthBasicProvider</a> <code>On Off provider-name [provider-name] ...</code>	On
<a href="#">AuthBasicUseDigestAlgorithm</a> <code>MD5 Off</code>	Off
Check passwords against the authentication providers as if Digest Authentication was in for Basic Authentication.	
<a href="#">AuthDBDUserPWQuery</a> <code>query</code>	
SQL query to look up a password for a user	
<a href="#">AuthDBDUserRealmQuery</a> <code>query</code>	
SQL query to look up a password hash for a user and realm.	
<a href="#">AuthDBMGroupFile</a> <code>file-path</code>	
<a href="#">AuthDBMType</a>	default
<code>default SDBM GDBM NDBM DB</code>	
<a href="#">AuthDBMUserFile</a> <code>file-path</code>	
<a href="#">AuthDigestAlgorithm</a> <code>MD5 MD5-sess</code>	MD5
digest authentication challenge response hash	
<a href="#">AuthDigestDomain</a> <code>URI [URI] ...</code>	
digest authentication URI	
<a href="#">AuthDigestNonceLifetime</a> <code>seconds</code>	300
nonce	
<a href="#">AuthDigestProvider</a> <code>On Off provider-name [provider-name] ...</code>	On
<a href="#">AuthDigestQop</a> <code>none auth auth-int [auth auth-int]</code>	auth
digest authentication (quality-of-protection) .	
<a href="#">AuthDigestShmemSize</a> <code>size</code>	1000
<a href="#">AuthFormAuthoritative</a> <code>On Off</code>	On
Sets whether authorization and authentication are passed to lower level modules	

<a href="#"><u>AuthFormBody <i>fieldname</i></u></a>	
The name of a form field carrying the body of the request to attempt on successful login	
<a href="#"><u>AuthFormDisableNoStore <i>On Off</i></u></a>	Off
Disable the CacheControl no-store header on the login page	
<a href="#"><u>AuthFormFakeBasicAuth <i>On Off</i></u></a>	Off
Fake a Basic Authentication header	
<a href="#"><u>AuthFormLocation <i>fieldname</i></u></a>	
The name of a form field carrying a URL to redirect to on successful login	
<a href="#"><u>AuthFormLoginRequiredLocation <i>url</i></u></a>	
The URL of the page to be redirected to should login be required	
<a href="#"><u>AuthFormLoginSuccessLocation <i>url</i></u></a>	
The URL of the page to be redirected to should login be successful	
<a href="#"><u>AuthFormLogoutLocation <i>uri</i></u></a>	
The URL to redirect to after a user has logged out	
<a href="#"><u>AuthFormMethod <i>fieldname</i></u></a>	
The name of a form field carrying the method of the request to attempt on successful login	
<a href="#"><u>AuthFormMimetype <i>fieldname</i></u></a>	
The name of a form field carrying the mimetype of the body of the request to attempt on suc	
<a href="#"><u>AuthFormPassword <i>fieldname</i></u></a>	
The name of a form field carrying the login password	
<a href="#"><u>AuthFormProvider <i>provider-name [provider-name] ...</i></u></a>	file
Sets the authentication provider(s) for this location	
<a href="#"><u>AuthFormSitePassphrase <i>secret</i></u></a>	
Bypass authentication checks for high traffic sites	
<a href="#"><u>AuthFormSize <i>size</i></u></a>	
The largest size of the form in bytes that will be parsed for the login details	
<a href="#"><u>AuthFormUsername <i>fieldname</i></u></a>	
The name of a form field carrying the login username	
<a href="#"><u>AuthGroupFile <i>file-path</i></u></a>	
<a href="#"><u>AuthLDAPAuthorizePrefix <i>prefix</i></u></a>	AUTHORIZE_
Specifies the prefix for environment variables set during authorization	
<a href="#"><u>AuthLDAPBindAuthoritative <i>off on</i></u></a>	on
Determines if other authentication providers are used when a user can be mapped to a DN	

cannot successfully bind with the user's credentials.	
<a href="#">AuthLDAPBindDN <i>distinguished-name</i></a>	
Optional DN to use in binding to the LDAP server	
<a href="#">AuthLDAPBindPassword <i>password</i></a>	
Password used in conjunction with the bind DN	
<a href="#">AuthLDAPCharsetConfig <i>file-path</i></a>	
Language to charset conversion configuration file	
<a href="#">AuthLDAPCompareAsUser on off</a>	off
Use the authenticated user's credentials to perform authorization comparisons	
<a href="#">AuthLDAPCompareDNOnServer on off</a>	on
Use the LDAP server to compare the DNS	
<a href="#">AuthLDAPDereferenceAliases <i>never searching finding always</i></a>	always
When will the module de-reference aliases	
<a href="#">AuthLDAPGroupAttribute <i>attribute</i></a>	member uniquememb
LDAP attributes used to identify the user members of groups.	
<a href="#">AuthLDAPGroupAttributeIsDN on off</a>	on
Use the DN of the client username when checking for group membership	
<a href="#">AuthLDAPInitialBindAsUser off on</a>	off
Determines if the server does the initial DN lookup using the basic authentication users' own instead of anonymously or with hard-coded credentials for the server	
<a href="#">AuthLDAPInitialBindPattern <i>regex substitution</i></a>	(.*) \$1 (remote use +
Specifies the transformation of the basic authentication username to be used when binding server to perform a DN lookup	
<a href="#">AuthLDAPMaxSubGroupDepth <i>Number</i></a>	10
Specifies the maximum sub-group nesting depth that will be evaluated before the user search	
<a href="#">AuthLDAPRemoteUserAttribute <i>uid</i></a>	
Use the value of the attribute returned during the user query to set the REMOTE_USER environment variable	
<a href="#">AuthLDAPRemoteUserIsDN on off</a>	off
Use the DN of the client username to set the REMOTE_USER environment variable	
<a href="#">AuthLDAPSearchAsUser on off</a>	off
Use the authenticated user's credentials to perform authorization searches	
<a href="#">AuthLDAPSubGroupAttribute <i>attribute</i></a>	
Specifies the attribute labels, one value per directive line, used to distinguish the members of a group that are groups.	
<a href="#">AuthLDAPSubGroupClass <i>LdapObjectClass</i></a>	groupOfNames groupOfNames

Specifies which LDAP objectClass values identify directory objects that are groups during search processing.	
<a href="#">AuthLDAPUrl <i>url [NONE SSL TLS STARTTLS]</i></a>	
URL specifying the LDAP search parameters	
<a href="#">AuthMerging <i>Off   And   Or</i></a>	Off
Controls the manner in which each configuration section's authorization logic is combined with preceding configuration sections.	
<a href="#">AuthName <i>auth-domain</i></a>	
Authorization realm for use in HTTP authentication	
<a href="#">AuthnCacheContext <i>directory server custom-string</i></a>	
Specify a context string for use in the cache key	
<a href="#">AuthnCacheEnable</a>	
Enable Authn caching configured anywhere	
<a href="#">AuthnCacheProvideFor <i>authn-provider [...]</i></a>	
Specify which authn provider(s) to cache for	
<a href="#">AuthnCacheSOCache <i>provider-name[:provider-args]</i></a>	
Select socache backend provider to use	
<a href="#">AuthnCacheTimeout <i>timeout (seconds)</i></a>	
Set a timeout for cache entries	
<a href="#">&lt;AuthnProviderAlias <i>baseProvider Alias</i>&gt; ...</a> <a href="#">&lt;/AuthnProviderAlias&gt;</a>	
Enclose a group of directives that represent an extension of a base authentication provider, the specified alias	
<a href="#">AuthnzFcgiCheckAuthnProvider <i>provider-name None option ...</i></a>	
Enables a FastCGI application to handle the check_authn authentication hook.	
<a href="#">AuthnzFcgiDefineProvider <i>type provider-name backend-address</i></a>	
Defines a FastCGI application as a provider for authentication and/or authorization	
<a href="#">AuthType <i>None Basic Digest Form</i></a>	
Type of user authentication	
<a href="#">AuthUserFile <i>file-path</i></a>	
<a href="#">AuthzDBDLoginToReferer <i>On Off</i></a>	Off

	Determines whether to redirect the Client to the Referring page on successful login or logou request header is present	
<a href="#">AuthzDBDQuery <i>query</i></a>	Specify the SQL Query for the required operation	
<a href="#">AuthzDBDRedirectQuery <i>query</i></a>	Specify a query to look up a login page for the user	
<a href="#">AuthzDBMType</a> <a href="#">default SDBM GDBM NDBM DB</a>		default
<a href="#">&lt;AuthzProviderAlias <i>baseProvider Alias Require-Parameters</i>&gt; ...</a> <a href="#">&lt;/AuthzProviderAlias&gt;</a>	Enclose a group of directives that represent an extension of a base authorization provider a the specified alias	
<a href="#">AuthzSendForbiddenOnFailure On Off</a>	Send '403 FORBIDDEN' instead of '401 UNAUTHORIZED' if authentication succeeds but a	Off
<a href="#">BalancerGrowth #</a>	Number of additional Balancers that can be added Post-configuration	5
<a href="#">BalancerInherit On Off</a>	Inherit ProxyPassed Balancers/Workers from the main server	On
<a href="#">BalancerMember [<i>balancerurl</i>] <i>url</i> [<i>key=value</i> [<i>key=value ...</i>]]</a>	Add a member to a load balancing group	
<a href="#">BalancerPersist On Off</a>	Attempt to persist changes made by the Balancer Manager across restarts.	Off
<a href="#">BrotliAlterETag AddSuffix NoChange Remove</a>	How the outgoing ETag header should be modified during compression	AddSuffix
<a href="#">BrotliCompressionMaxInputBlock <i>value</i></a>	Maximum input block size	
<a href="#">BrotliCompressionQuality <i>value</i></a>	Compression quality	5
<a href="#">BrotliCompressionWindow <i>value</i></a>	Brotli sliding compression window size	18
<a href="#">BrotliFilterNote [<i>type</i>] <i>notename</i></a>	Places the compression ratio in a note for logging	

<a href="#">BrowserMatch <i>regex</i> [!]env-variable[=value] [!]env-variable[=value] ...</a>	
HTTP User-Agent	
<a href="#">BrowserMatchNoCase <i>regex</i> [!]env-variable[=value] [!]env-variable[=value] ...</a>	
User-Agent	
Buffer log entries in memory before writing to disk	
<a href="#">BufferSize <i>integer</i></a>	131072
Maximum size in bytes to buffer by the buffer filter	
<a href="#">CacheDefaultExpire <i>seconds</i></a>	3600 (one hour)
.	
Add an X-Cache-Detail header to the response.	
<a href="#">CacheDirLength <i>length</i></a>	2
<a href="#">CacheDirLevels <i>levels</i></a>	3
.	
<a href="#">CacheDisable <i>url-string</i></a>	
URL	
<a href="#">CacheEnable <i>cache_type url-string</i></a>	
URL	
<a href="#">CacheFile <i>file-path</i> [file-path] ...</a>	
.	
Add an X-Cache header to the response.	
<a href="#">CacheIgnoreCacheControl <i>On Off</i></a>	Off
.	
<a href="#">CacheIgnoreHeaders <i>header-string</i> [header-string] ...</a>	None
HTTP ()	
<a href="#">CacheIgnoreNoLastMod <i>On Off</i></a>	Off
Last Modified .	
Ignore query string when caching	

Ignore defined session identifiers encoded in the URL when caching	
Override the base URL of reverse proxied cache keys.	
<a href="#"><u>CacheLastModifiedFactor</u></a> <i>float</i>	0.1
LastModified .	
Enable the thundering herd lock.	
Set the maximum possible age of a cache lock.	
Set the lock path directory.	
<a href="#"><u>CacheMaxExpire</u></a> <i>seconds</i>	86400 ()
<a href="#"><u>CacheMaxFileSize</u></a> <i>bytes</i>	1000000
()	
The minimum time in seconds to cache a document	
<a href="#"><u>CacheMinFileSize</u></a> <i>bytes</i>	1
()	
<a href="#"><u>CacheNegotiatedDocs</u></a> On Off	Off
Allows content-negotiated documents to be cached by proxy servers	
Run the cache from the quick handler.	
The minimum size (in bytes) of the document to read and be cached before sending the dat	
The minimum time (in milliseconds) that should elapse while reading before data is sent do	
<a href="#"><u>CacheRoot</u></a> <i>directory</i>	
root	
<a href="#"><u>CacheSocache</u></a> <i>type[:args]</i>	
The shared object cache implementation to use	
<a href="#"><u>CacheSocacheMaxSize</u></a> <i>bytes</i>	102400
The maximum size (in bytes) of an entry to be placed in the cache	
<a href="#"><u>CacheSocacheMaxTime</u></a> <i>seconds</i>	86400

The maximum time (in seconds) for a document to be placed in the cache	
<a href="#"><u>CacheSocacheMinTime <i>seconds</i></u></a>	600
The minimum time (in seconds) for a document to be placed in the cache	
<a href="#"><u>CacheSocacheReadSize <i>bytes</i></u></a>	0
The minimum size (in bytes) of the document to read and be cached before sending the data	
<a href="#"><u>CacheSocacheReadTime <i>milliseconds</i></u></a>	0
The minimum time (in milliseconds) that should elapse while reading before data is sent down	
Serve stale content in place of 5xx responses.	
Attempt to cache responses that the server reports as expired	
Attempt to cache requests or responses that have been marked as no-store.	
Attempt to cache responses that the server has marked as private	
<a href="#"><u>CGIDScriptTimeout <i>time[s ms]</i></u></a>	
The length of time to wait for more output from the CGI program	
<a href="#"><u>CGIMapExtension <i>cgi-path .extension</i></u></a>	
Technique for locating the interpreter for CGI scripts	
<a href="#"><u>CGIPassAuth On Off</u></a>	Off
Enables passing HTTP authorization headers to scripts as CGI variables	
<a href="#"><u>CGIVar <i>variable rule</i></u></a>	
Controls how some CGI variables are set	
<a href="#"><u>CharsetDefault <i>charset</i></u></a>	
<a href="#"><u>CharsetOptions <i>option [option] ...</i></u></a>	DebugLevel=0 NoImp
<a href="#"><u>CharsetSourceEnc <i>charset</i></u></a>	
Limits the action of the spelling module to case corrections	
<a href="#"><u>CheckSpelling on off</u></a>	Off
<a href="#"><u>ChrootDir <i>/path/to/directory</i></u></a>	
Directory for apache to run chroot(8) after startup.	

<a href="#">ContentDigest On Off</a>	Off
Enables the generation of Content - MD5 HTTP Response headers	
<a href="#">CookieDomain <i>domain</i></a>	
The domain to which the tracking cookie applies	
<a href="#">CookieExpires <i>expiry-period</i></a>	
Expiry time for the tracking cookie	
<a href="#">CookieName <i>token</i></a>	Apache
Name of the tracking cookie	
<a href="#">CookieStyle</a>	Netscape
<a href="#">Netscape Cookie Cookie2 RFC2109 RFC2965</a>	
Format of the cookie header field	
<a href="#">CookieTracking on off</a>	off
Enables tracking cookie	
<a href="#">CoreDumpDirectory <i>directory</i></a>	
Directory where Apache HTTP Server attempts to switch before dumping core	
<a href="#">CustomLog <i>file pipe format nickname [env=</i></a> <a href="#">[!]<i>environment-variable</i></a>	
<a href="#">Dav On Off <i>provider-name</i></a>	Off
WebDAV HTTP	
<a href="#">DavDepthInfinity on off</a>	off
PROPFIND Depth: Infinity	
<a href="#">DavGenericLockDB <i>file-path</i></a>	
Location of the DAV lock database	
<a href="#">DavLockDB <i>file-path</i></a>	
DAV	
<a href="#">DavMinTimeout <i>seconds</i></a>	0
DAV	
<a href="#">DBDExptime <i>time-in-seconds</i></a>	300
Keepalive time for idle connections	
<a href="#">DBDInitSQL "<i>SQL statement</i>"</a>	
Execute an SQL statement after connecting to a database	
<a href="#">DBDKeep <i>number</i></a>	2
Maximum sustained number of connections	
<a href="#">DBDMax <i>number</i></a>	10

Maximum number of connections	
<a href="#">DBDMin <i>number</i></a>	1
Minimum number of connections	
<a href="#">DBDParams <i>param1=value1[,param2=value2]</i></a>	
Parameters for database connection	
<a href="#">DBDPersist On Off</a>	
Whether to use persistent connections	
<a href="#">DBDPrepareSQL "<i>SQL statement</i>" <i>label</i></a>	
Define an SQL prepared statement	
<a href="#">DBDriver <i>name</i></a>	
Specify an SQL driver	
<a href="#">DefaultIcon <i>url-path</i></a>	
<a href="#">DefaultLanguage <i>language-tag</i></a>	
Defines a default language-tag to be sent in the Content-Language header field for all resource context that have not been assigned a language-tag by some other means.	
<a href="#">DefaultRuntimeDir <i>directory-path</i></a>	DEFAULT_REL_RUN +
Base directory for the server run-time files	
<a href="#">DefaultType <i>media-type none</i></a>	none
This directive has no effect other than to emit warnings if the value is not none. In prior versions would specify a default media type to assign to response content for which no other media type could be found.	
<a href="#">Define <i>parameter-name</i> [<i>parameter-value</i>]</a>	
Define a variable	
<a href="#">DeflateBufferSize <i>value</i></a>	8096
zlib	
<a href="#">DeflateCompressionLevel <i>value</i></a>	
<a href="#">DeflateFilterNote [<i>type</i>] <i>notename</i></a>	
<a href="#">DeflateInflateLimitRequestBody <i>value</i></a>	
Maximum size of inflated request bodies	
<a href="#">DeflateInflateRatioBurst <i>value</i></a>	
Maximum number of times the inflation ratio for request bodies can be crossed	
<a href="#">DeflateInflateRatioLimit <i>value</i></a>	

Maximum inflation ratio for request bodies	
<a href="#">DeflateMemLevel <i>value</i></a>	9
zlib	
<a href="#">DeflateWindowSize <i>value</i></a>	15
Zlib window size	
<a href="#">Deny from all <i>host</i> env=[!]<i>env-variable</i> [<i>host</i> env=[!]<i>env-variable</i>] ...</a>	
Controls which hosts are denied access to the server	
<a href="#">&lt;Directory <i>directory-path</i>&gt; ... &lt;/Directory&gt;</a>	
Enclose a group of directives that apply only to the named file-system directory, sub-directory contents.	
<a href="#">DirectoryCheckHandler On Off</a>	Off
Toggle how this module responds when another handler is configured	
<a href="#">DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...</a>	index.html
<a href="#">DirectoryIndexRedirect on   off   permanent   temp   seeother   <i>3xx-code</i></a>	off
Configures an external redirect for directory indexes.	
<a href="#">&lt;DirectoryMatch <i>regex</i>&gt; ... &lt;/DirectoryMatch&gt;</a>	
Enclose directives that apply to the contents of file-system directories matching a regular expression	
<a href="#">DirectorySlash On Off</a>	On
<a href="#">DocumentRoot <i>directory-path</i></a>	"/usr/local/apache/ +
Directory that forms the main document tree visible from the web	
<a href="#">DTracePrivileges On Off</a>	Off
Determines whether the privileges required by dtrace are enabled.	
<a href="#">DumpIOInput On Off</a>	Off
Dump all input data to the error log	
<a href="#">DumpIOOutput On Off</a>	Off
Dump all output data to the error log	
<a href="#">&lt;Else&gt; ... &lt;/Else&gt;</a>	
Contains directives that apply only if the condition of a previous <a href="#">&lt;If&gt;</a> or <a href="#">&lt;ElseIf&gt;</a> section is satisfied by a request at runtime	
<a href="#">&lt;Elseif <i>expression</i>&gt; ... &lt;/Elseif&gt;</a>	
Contains directives that apply only if a condition is satisfied by a request at runtime while the previous <a href="#">&lt;If&gt;</a> or <a href="#">&lt;ElseIf&gt;</a> section is not satisfied	

<a href="#">EnableExceptionHook On Off</a>	Off
Enables a hook that runs exception handlers after a crash	
<a href="#">EnableMMAP On Off</a>	On
Use memory-mapping to read files during delivery	
<a href="#">EnableSendfile On Off</a>	Off
Use the kernel sendfile support to deliver files to the client	
<a href="#">Error message</a>	
Abort configuration parsing with a custom error message	
<a href="#">ErrorDocument <i>error-code</i> <i>document</i></a>	
What the server will return to the client in case of an error	
<a href="#">ErrorLog <i>file-path</i> syslog[:<i>[facility]</i>[:<i>tag</i>]]</a>	logs/error_log (Uni +
Location where the server will log errors	
<a href="#">ErrorLogFormat [<i>connection</i> <i>request</i>] <i>format</i></a>	
Format specification for error log entries	
<a href="#">Example</a>	
API	
<a href="#">ExpiresActive On Off</a>	
Expires	
<a href="#">ExpiresByType <i>MIME-type</i> <i>&lt;code&gt;seconds</i></a>	
MIME type Expires	
<a href="#">ExpiresDefault <i>&lt;code&gt;seconds</i></a>	
<a href="#">ExtendedStatus On Off</a>	Off[*]
Keep track of extended status information for each request	
<a href="#">ExtFilterDefine <i>filtername</i> <i>parameters</i></a>	
<a href="#">ExtFilterOptions <i>option</i> [<i>option</i>] ...</a>	DebugLevel=0 NoLog
<a href="#">mod_ext_filter</a>	
Define a default URL for requests that don't map to a file	
<a href="#">FileETag <i>component</i> ...</a>	MTime Size
File attributes used to create the ETag HTTP response header for static files	
<a href="#">&lt;Files <i>filename</i>&gt; ... &lt;/Files&gt;</a>	
Contains directives that apply to matched filenames	
<a href="#">&lt;FilesMatch <i>regex</i>&gt; ... &lt;/FilesMatch&gt;</a>	

Contains directives that apply to regular-expression matched filenames	
<a href="#">FilterChain [+=-@!]filter-name ...</a>	
Configure the filter chain	
<a href="#">FilterDeclare filter-name [type]</a>	
Declare a smart filter	
<a href="#">FilterProtocol filter-name [provider-name] proto-flags</a>	
Deal with correct HTTP protocol handling	
<a href="#">FilterProvider filter-name provider-name expression</a>	
Register a content filter	
<a href="#">FilterTrace filter-name level</a>	
Get debug/diagnostic information from <code>mod_filter</code>	
<a href="#">ForceLanguagePriority None Prefer Fallback [Prefer Fallback]</a>	Prefer
Action to take if a single acceptable document is not found	
<a href="#">ForceType media-type None</a>	
Forces all matching files to be served with the specified media type in the HTTP Content-Type	
<a href="#">ForensicLog filename pipe</a>	
Sets filename of the forensic log	
<a href="#">GlobalLogfile pipe format nickname [env=[!]environment-variable  expr=expression]</a>	
Sets filename and format of log file	
<a href="#">GprofDir /tmp/gprof/ /tmp/gprof/%</a>	
Directory to write gmon.out profiling data to.	
<a href="#">GracefulShutdownTimeout seconds</a>	0
Specify a timeout after which a gracefully shutdown server will exit.	
<a href="#">Group unix-group</a>	#-1
Group under which the server will answer requests	
<a href="#">H2CopyFiles on off</a>	off
Determine file handling in responses	
<a href="#">H2Direct on off</a>	on for h2c, off for +
H2 Direct Protocol Switch	
<a href="#">H2EarlyHints on off</a>	off
Determine sending of 103 status codes	

<a href="#">H2MaxSessionStreams <i>n</i></a>	100
Maximum number of active streams per HTTP/2 session.	
<a href="#">H2MaxWorkerIdleSeconds <i>n</i></a>	600
Maximum number of seconds h2 workers remain idle until shut down.	
<a href="#">H2MaxWorkers <i>n</i></a>	
Maximum number of worker threads to use per child process.	
<a href="#">H2MinWorkers <i>n</i></a>	
Minimal number of worker threads to use per child process.	
<a href="#">H2ModernTLSOnly on off</a>	on
Require HTTP/2 connections to be "modern TLS" only	
<a href="#">H2Push on off</a>	on
H2 Server Push Switch	
<a href="#">H2PushDiarySize <i>n</i></a>	256
H2 Server Push Diary Size	
<a href="#">H2PushPriority <i>mime-type</i></a> <a href="#">[after before interleaved]</a> <a href="#">[weight]</a>	* After 16
H2 Server Push Priority	
<a href="#">H2PushResource [add] path [critical]</a>	
Declares resources for early pushing to the client	
<a href="#">H2SerializeHeaders on off</a>	off
Serialize Request/Response Processing Switch	
<a href="#">H2StreamMaxMemSize <i>bytes</i></a>	65536
Maximum amount of output data buffered per stream.	
<a href="#">H2TLSCoolDownSecs <i>seconds</i></a>	1
-	
<a href="#">H2TLWarmUpSize <i>amount</i></a>	1048576
-	
<a href="#">H2Upgrade on off</a>	on for h2c, off for +
H2 Upgrade Protocol Switch	
<a href="#">H2WindowSize <i>bytes</i></a>	65535
Size of Stream Window for upstream data.	
<a href="#">Header [condition] set append add unset echo</a> <a href="#">header [value] [early env=[!]<i>variable</i>]</a>	
HTTP	
<a href="#">HeaderName <i>filename</i></a>	

<a href="#">HeartbeatAddress <i>addr:port</i></a>	
Multicast address for heartbeat packets	
<a href="#">HeartbeatListen<i>addr:port</i></a>	
multicast address to listen for incoming heartbeat requests	
<a href="#">HeartbeatMaxServers <i>number-of-servers</i></a>	10
Specifies the maximum number of servers that will be sending heartbeat requests to this se	
<a href="#">HeartbeatStorage <i>file-path</i></a>	logs/hb.dat
Path to store heartbeat data	
<a href="#">HeartbeatStorage <i>file-path</i></a>	logs/hb.dat
Path to read heartbeat data	
<a href="#">HostnameLookups On Off Double</a>	Off
Enables DNS lookups on client IP addresses	
<a href="#">HttpProtocolOptions [Strict Unsafe] [RegisteredMethods LenientMethods] [Allow0.9 Require1.0]</a>	Strict LenientMetho +
Modify restrictions on HTTP Request Messages	
<a href="#">IdentityCheck On Off</a>	Off
RFC 1413	
<a href="#">IdentityCheckTimeout <i>seconds</i></a>	30
ident	
<a href="#">&lt;If <i>expression</i>&gt; ... &lt;/If&gt;</a>	
Contains directives that apply only if a condition is satisfied by a request at runtime	
<a href="#">&lt;IfDefine [!]<i>parameter-name</i>&gt; ... &lt;/IfDefine&gt;</a>	
Encloses directives that will be processed only if a test is true at startup	
<a href="#">&lt;IfModule [!]<i>module-file module-identifier</i>&gt; ... &lt;/IfModule&gt;</a>	
Encloses directives that are processed conditional on the presence or absence of a specific	
<a href="#">&lt;IfVersion [!]<i>operator</i> <i>version</i>&gt; ... &lt;/IfVersion&gt;</a>	
<a href="#">ImapBase <i>map referer URL</i></a>	http://servername/
base	
<a href="#">ImapDefault <i>error nocontent map referer URL</i></a>	nocontent

<a href="#">ImapMenu</a> <a href="#">none formatted semiformatted unformatted</a>	
<a href="#">Include <i>file-path directory-path wildcard</i></a> Includes other configuration files from within the server configuration files	
<a href="#">IncludeOptional <i>file-path directory-path wildcard</i></a> Includes other configuration files from within the server configuration files	
Inserts text in the HEAD section of an index page.	
<a href="#">IndexIgnore <i>file [file] ...</i></a>	
<a href="#">IndexIgnoreReset ON OFF</a> Empties the list of files to hide when listing a directory	
<a href="#">IndexOptions <i>[+ -]option [[+ -]option] ...</i></a>	
<a href="#">IndexOrderDefault <i>Ascending Descending Name Date Size Description</i></a>	Ascending Name
<a href="#">IndexStyleSheet <i>url-path</i></a> CSS	
<a href="#">InputSed <i>sed-command</i></a> Sed command to filter request data (typically POST data)	
<a href="#">ISAPIAppendLogToErrors <i>on off</i></a> ISAPI exntension HSE_APPEND_LOG_PARAMETER	off
<a href="#">ISAPIAppendLogToQuery <i>on off</i></a> ISAPI exntension HSE_APPEND_LOG_PARAMETER	on
<a href="#">ISAPICacheFile <i>file-path [file-path] ...</i></a> ISAPI .dll	
<a href="#">ISAPIFakeAsync <i>on off</i></a> ISAPI	off
<a href="#">ISAPILogNotSupported <i>on off</i></a> ISAPI extension	off
<a href="#">ISAPIReadAheadBuffer <i>size</i></a> ISAPI extension (read ahead buffer)	49152

<a href="#">KeepAlive On Off</a>	On
Enables HTTP persistent connections	
<a href="#">KeepAliveTimeout <i>num</i>[ms]</a>	5
Amount of time the server will wait for subsequent requests on a persistent connection	
<a href="#">KeptBodySize <i>maximum size in bytes</i></a>	0
Keep the request body instead of discarding it up to the specified maximum size, for potenti such as mod_include.	
<a href="#">LanguagePriority <i>MIME-lang</i> [<i>MIME-lang</i>] ...</a>	
The precedence of language variants for cases where the client does not express a preferene	
<a href="#">LDAPCacheEntries <i>number</i></a>	1024
Maximum number of entries in the primary LDAP cache	
<a href="#">LDAPCacheTTL <i>seconds</i></a>	600
Time that cached items remain valid	
<a href="#">LDAPConnectionPoolTTL <i>n</i></a>	-1
Discard backend connections that have been sitting in the connection pool too long	
<a href="#">LDAPConnectionTimeout <i>seconds</i></a>	
Specifies the socket connection timeout in seconds	
<a href="#">LDAPLibraryDebug <i>7</i></a>	
Enable debugging in the LDAP SDK	
<a href="#">LDAPOpCacheEntries <i>number</i></a>	1024
Number of entries used to cache LDAP compare operations	
<a href="#">LDAPOpCacheTTL <i>seconds</i></a>	600
Time that entries in the operation cache remain valid	
<a href="#">LDAPReferralHopLimit <i>number</i></a>	
The maximum number of referral hops to chase before terminating an LDAP query.	
<a href="#">LDAPReferrals <i>On Off default</i></a>	On
Enable referral chasing during queries to the LDAP server.	
<a href="#">LDAPRetries <i>number-of-retries</i></a>	3
Configures the number of LDAP server retries.	
<a href="#">LDAPRetryDelay <i>seconds</i></a>	0
Configures the delay between LDAP server retries.	
<a href="#">LDAPSharedCacheFile <i>directory-path/filename</i></a>	
Sets the shared memory cache file	
<a href="#">LDAPSharedCacheSize <i>bytes</i></a>	500000

Size in bytes of the shared-memory cache	
<a href="#"><u>LDAPTimeout <i>seconds</i></u></a>	60
Specifies the timeout for LDAP search and bind operations, in seconds	
<a href="#"><u>LDAPTrustedClientCert <i>type directory-path/filename/nickname [password]</i></u></a>	
Sets the file containing or nickname referring to a per connection client certificate. Not all LL support per connection client certificates.	
<a href="#"><u>LDAPTrustedGlobalCert <i>type directory-path/filename [password]</i></u></a>	
Sets the file or database containing global trusted Certificate Authority or global client certifi	
<a href="#"><u>LDAPTrustedMode <i>type</i></u></a>	
Specifies the SSL/TLS mode to be used when connecting to an LDAP server.	
<a href="#"><u>LDAPVerifyServerCert <i>On Off</i></u></a>	On
Force server certificate verification	
<a href="#"><u>&lt;Limit <i>method [method] ... &gt; ... &lt;/Limit&gt;</i></u></a>	
Restrict enclosed access controls to only certain HTTP methods	
<a href="#"><u>&lt;LimitExcept <i>method [method] ... &gt; ... &lt;/LimitExcept&gt;</i></u></a>	
Restrict access controls to all HTTP methods except the named ones	
<a href="#"><u>LimitInternalRecursion <i>number [number]</i></u></a>	10
Determine maximum number of internal redirects and nested subrequests	
<a href="#"><u>LimitRequestBody <i>bytes</i></u></a>	0
Restricts the total size of the HTTP request body sent from the client	
<a href="#"><u>LimitRequestFields <i>number</i></u></a>	100
Limits the number of HTTP request header fields that will be accepted from the client	
<a href="#"><u>LimitRequestFieldSize <i>bytes</i></u></a>	8190
Limits the size of the HTTP request header allowed from the client	
<a href="#"><u>LimitRequestLine <i>bytes</i></u></a>	8190
Limit the size of the HTTP request line that will be accepted from the client	
<a href="#"><u>LimitXMLRequestBody <i>bytes</i></u></a>	1000000
Limits the size of an XML-based request body	
<a href="#"><u>Listen [<i>IP-address:]portnumber [protocol]</i></u></a>	
IP addresses and ports that the server listens to	
<a href="#"><u>ListenBacklog <i>backlog</i></u></a>	
Maximum length of the queue of pending connections	

<a href="#">ListenCoresBucketsRatio</a> <i>ratio</i>	0 (disabled)
Ratio between the number of CPU cores (online) and the number of listeners' buckets	
<a href="#">LoadFile</a> <i>filename</i> [ <i>filename</i> ] ...	
<a href="#">LoadModule</a> <i>module filename</i>	
<a href="#">&lt;Location</a> <i>URL-path URL</i> > ... </Location>	
Applies the enclosed directives only to matching URLs	
<a href="#">&lt;LocationMatch</a> <i>regex</i> > ... </LocationMatch>	
Applies the enclosed directives only to regular-expression matching URLs	
<a href="#">LogFormat</a> <i>format nickname</i> [ <i>nickname</i> ]	"%h %l %u %t \"%r\" -
<a href="#">LogIOTrackTTFB</a> ON OFF	OFF
Enable tracking of time to first byte (TTFB)	
<a href="#">LogLevel</a> [ <i>module:</i> ] <i>level</i> [ <i>module:level</i> ] ...	warn
Controls the verbosity of the ErrorLog	
<a href="#">LogMessage</a> <i>message</i> [ <i>hook=hook</i> ] [ <i>expr=expression</i> ]	
Log user-defined message to error log	
<a href="#">LuaAuthzProvider</a> <i>provider_name</i> <i>/path/to/lua/script.lua</i> <i>function_name</i>	
Plug an authorization provider function into <code>mod_authz_core</code>	
<a href="#">LuaCodeCache</a> <i>stat forever never</i>	stat
Configure the compiled code cache.	
<a href="#">LuaHookAccessChecker</a> <i>/path/to/lua/script.lua</i> <i>hook_function_name</i> [ <i>early late</i> ]	
Provide a hook for the access_checker phase of request processing	
<a href="#">LuaHookAuthChecker</a> <i>/path/to/lua/script.lua</i> <i>hook_function_name</i> [ <i>early late</i> ]	
Provide a hook for the auth_checker phase of request processing	
<a href="#">LuaHookCheckUserID</a> <i>/path/to/lua/script.lua</i> <i>hook_function_name</i> [ <i>early late</i> ]	
Provide a hook for the check_user_id phase of request processing	
<a href="#">LuaHookFixups</a> <i>/path/to/lua/script.lua</i> <i>hook_function_name</i>	

Provide a hook for the fixups phase of a request processing	
<a href="#">LuaHookInsertFilter /path/to/lua/script.lua</a> <a href="#">hook_function_name</a>	
Provide a hook for the insert_filter phase of request processing	
<a href="#">LuaHookLog /path/to/lua/script.lua</a> <a href="#">log_function_name</a>	
Provide a hook for the access log phase of a request processing	
<a href="#">LuaHookMapToStorage /path/to/lua/script.lua</a> <a href="#">hook_function_name</a>	
Provide a hook for the map_to_storage phase of request processing	
<a href="#">LuaHookTranslateName /path/to/lua/script.lua</a> <a href="#">hook_function_name [early late]</a>	
Provide a hook for the translate name phase of request processing	
<a href="#">LuaHookTypeChecker /path/to/lua/script.lua</a> <a href="#">hook_function_name</a>	
Provide a hook for the type_checker phase of request processing	
<a href="#">LuaInherit none parent-first parent-last</a>	parent-first
Controls how parent configuration sections are merged into children	
<a href="#">LuaInputFilter filter_name</a> <a href="#">/path/to/lua/script.lua function_name</a>	
Provide a Lua function for content input filtering	
<a href="#">LuaMapHandler uri-pattern</a> <a href="#">/path/to/lua/script.lua [function-name]</a>	
Map a path to a lua handler	
<a href="#">LuaOutputFilter filter_name</a> <a href="#">/path/to/lua/script.lua function_name</a>	
Provide a Lua function for content output filtering	
<a href="#">LuaPackageCPath /path/to/include/?.soa</a>	
Add a directory to lua's package.cpath	
<a href="#">LuaPackagePath /path/to/include/?.lua</a>	
Add a directory to lua's package.path	
<a href="#">LuaQuickHandler /path/to/script.lua</a> <a href="#">hook_function_name</a>	
Provide a hook for the quick handler of request processing	
<a href="#">LuaRoot /path/to/a/directory</a>	

Specify the base path for resolving relative paths for mod_lua directives	
<a href="#">LuaScope</a> <i>once request conn thread server</i> <i>[min] [max]</i>	once
One of once, request, conn, thread -- default is once	
<a href="#">&lt;Macro name [par1 .. parN]&gt; ... &lt;/Macro&gt;</a>	
Define a configuration file macro	
<a href="#">MaxConnectionsPerChild</a> <i>number</i>	0
Limit on the number of connections that an individual child server will handle during its life	
<a href="#">MaxKeepAliveRequests</a> <i>number</i>	100
Number of requests allowed on a persistent connection	
<a href="#">MaxMemFree</a> <i>KBytes</i>	2048
Maximum amount of memory that the main allocator is allowed to hold without calling free	
<a href="#">MaxRangeOverlaps</a> <i>default   unlimited   none  </i> <i>number-of-ranges</i>	20
Number of overlapping ranges (eg: 100-200, 150-300) allowed before returning the comp	
<a href="#">MaxRangeReversals</a> <i>default   unlimited   none</i> <i>  number-of-ranges</i>	20
Number of range reversals (eg: 100-200, 50-70) allowed before returning the complete re	
<a href="#">MaxRanges</a> <i>default   unlimited   none  </i> <i>number-of-ranges</i>	200
Number of ranges allowed before returning the complete resource	
<a href="#">MaxRequestWorkers</a> <i>number</i>	
Maximum number of connections that will be processed simultaneously	
<a href="#">MaxSpareServers</a> <i>number</i>	10
Maximum number of idle child server processes	
<a href="#">MaxSpareThreads</a> <i>number</i>	
Maximum number of idle threads	
<a href="#">MaxThreads</a> <i>number</i>	2048
Set the maximum number of worker threads	
<a href="#">MemcacheConnTTL</a> <i>num[units]</i>	15s
Keepalive time for idle connections	
<a href="#">MergeTrailers</a> <i>[on off]</i>	off
Determines whether trailers are merged into headers	
<a href="#">MetaDir</a> <i>directory</i>	.web
CERN	

<a href="#">MetaFiles on off</a> CERN	off
<a href="#">MetaSuffix <i>suffix</i></a> CERN	.meta
<a href="#">MimeMagicFile <i>file-path</i></a> Enable MIME-type determination based on file contents using the specified magic file	
<a href="#">MinSpareServers <i>number</i></a> Minimum number of idle child server processes	5
<a href="#">MinSpareThreads <i>number</i></a> Minimum number of idle threads available to handle request spikes	
<a href="#">MMapFile <i>file-path</i> [<i>file-path</i>] ...</a>	
<a href="#">ModemStandard V.21 V.26bis V.32 V.34 V.92</a> Modem standard to simulate	
<a href="#">ModMimeUsePathInfo On Off</a> Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename	Off
<a href="#">MultiviewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters]</a> The types of files that will be included when searching for a matching file with MultiViews	NegotiatedOnly
<a href="#">Mutex <i>mechanism</i> [default <i>mutex-name</i>] ... [OmitPID]</a> Configures mutex mechanism and lock file directory for all or specified mutexes	default
<a href="#">NameVirtualHost <i>addr</i>[:<i>port</i>]</a> DEPRECATED: Designates an IP address for name-virtual hosting	
<a href="#">NoProxy <i>host</i> [<i>host</i>] ...</a> Hosts, domains, or networks that will be connected to directly	
<a href="#">NWSSLTrustedCerts <i>filename</i> [<i>filename</i>] ...</a> List of additional client certificates	
<a href="#">NWSSLUpgradeable [<i>IP-address</i>:]<i>portnumber</i></a> Allows a connection to be upgraded to an SSL connection upon request	
<a href="#">Options [+ -]<i>option</i> [[+ -]<i>option</i>] ...</a> Configures what features are available in a particular directory	FollowSymlinks
<a href="#">Order <i>ordering</i></a> Controls the default access state and the order in which <code>Allow</code> and <code>Deny</code> are evaluated.	Deny,Allow

<a href="#">OutputSed <i>sed-command</i></a>	
Sed command for filtering response content	
<a href="#">PassEnv <i>env-variable</i> [<i>env-variable</i>] ...</a>	
<a href="#">PidFile <i>filename</i></a>	logs/httpd.pid
File where the server records the process ID of the daemon	
<a href="#">PrivilegesMode FAST SECURE SELECTIVE</a>	FAST
Trade off processing speed and efficiency vs security against malicious privileges-aware co	
<a href="#">Protocol <i>protocol</i></a>	
Protocol for a listening socket	
<a href="#">ProtocolEcho On Off</a>	
echo	
<a href="#">Protocols <i>protocol</i> ...</a>	http/1.1
Protocols available for a server/virtual host	
<a href="#">ProtocolsHonorOrder On Off</a>	On
Determines if order of Protocols determines precedence during negotiation	
<a href="#">&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</a>	
Container for directives applied to proxied resources	
<a href="#">ProxyAddHeaders Off On</a>	On
Add proxy information in X-Forwarded-* headers	
<a href="#">ProxyBadHeader IsError Ignore StartBody</a>	IsError
Determines how to handle bad header lines in a response	
<a href="#">ProxyBlock * <i>word</i> <i>host</i> <i>domain</i></a>	
<a href="#">[<i>word</i> <i>host</i> <i>domain</i>] ...</a>	
Words, hosts, or domains that are banned from being proxied	
<a href="#">ProxyDomain <i>Domain</i></a>	
Default domain name for proxied requests	
<a href="#">ProxyErrorOverride On Off</a>	Off
Override error pages for proxied content	
<a href="#">ProxyExpressDBMFile &lt;pathname&gt;</a>	
Pathname to DBM file.	
<a href="#">ProxyExpressDBMFile &lt;type&gt;</a>	
DBM type of file.	
<a href="#">ProxyExpressEnable [on off]</a>	
Enable the module functionality.	

<a href="#">ProxyFCGIBackendType</a> <i>FPM GENERIC</i>	FPM
Specify the type of backend FastCGI application	
<a href="#">ProxyFCGISetEnvIf</a> <i>conditional-expression</i> <a href="#">[!]</a> <i>environment-variable-name</i> <a href="#">[value-expression]</a>	
Allow variables sent to FastCGI servers to be fixed up	
<a href="#">ProxyFtpDirCharset</a> <i>character set</i>	ISO-8859-1
Define the character set for proxied FTP listings	
<a href="#">ProxyFtpEscapeWildcards</a> <i>[on off]</i>	
Whether wildcards in requested filenames are escaped when sent to the FTP server	
<a href="#">ProxyFtpListOnWildcard</a> <i>[on off]</i>	
Whether wildcards in requested filenames trigger a file listing	
<a href="#">ProxyHCE</a> <i>Expr name {ap_expr expression}</i>	
Creates a named condition expression to use to determine health of the backend based on	
<a href="#">ProxyHCTemplate</a> <i>name parameter=setting</i> <i>&lt;...&gt;</i>	
Creates a named template for setting various health check parameters	
<a href="#">ProxyHCTPsize</a> <i>&lt;size&gt;</i>	
Sets the total server-wide size of the threadpool used for the health check workers.	
<a href="#">ProxyHTMLBufSize</a> <i>bytes</i>	
Sets the buffer size increment for buffering inline scripts and stylesheets.	
<a href="#">ProxyHTMLCharsetOut</a> <i>Charset   *</i>	
Specify a charset for mod_proxy_html output.	
<a href="#">ProxyHTMLDocType</a> <i>HTML XHTML [Legacy]</i>	
<b>OR</b>	
<a href="#">ProxyHTMLDocType</a> <i>fpi [SGML XML]</i>	
Sets an HTML or XHTML document type declaration.	
<a href="#">ProxyHTMLEnable</a> <i>On Off</i>	Off
Turns the proxy_html filter on or off.	
<a href="#">ProxyHTMLEvents</a> <i>attribute [attribute ...]</i>	
Specify attributes to treat as scripting events.	
<a href="#">ProxyHTMLExtended</a> <i>On Off</i>	Off
Determines whether to fix links in inline scripts, stylesheets, and scripting events.	
<a href="#">ProxyHTMLFixups</a> <i>[lowercase] [dospath]</i> <i>[reset]</i>	

Fixes for simple HTML errors.	
<a href="#">ProxyHTMLInterp <i>On Off</i></a>	Off
Enables per-request interpolation of <a href="#">ProxyHTMLURLMap</a> rules.	
<a href="#">ProxyHTMLLinks <i>element attribute [attribute2 ...]</i></a>	
Specify HTML elements that have URL attributes to be rewritten.	
<a href="#">ProxyHTMLMeta <i>On Off</i></a>	Off
Turns on or off extra pre-parsing of metadata in HTML <head> sections.	
<a href="#">ProxyHTMLStripComments <i>On Off</i></a>	Off
Determines whether to strip HTML comments.	
<a href="#">ProxyHTMLURLMap <i>from-pattern to-pattern [flags] [cond]</i></a>	
Defines a rule to rewrite HTML links	
<a href="#">ProxyIOBufferSize <i>bytes</i></a>	8192
Determine size of internal data throughput buffer	
<a href="#">&lt;ProxyMatch <i>regex</i>&gt; ...&lt;/ProxyMatch&gt;</a>	
Container for directives applied to regular-expression-matched proxied resources	
<a href="#">ProxyMaxForwards <i>number</i></a>	-1
Maximum number of proxies that a request can be forwarded through	
<a href="#">ProxyPass <i>[path] ! url [key=value [key=value ...]] [nocanon] [interpolate] [noquery]</i></a>	
Maps remote servers into the local server URL-space	
<a href="#">ProxyPassInherit <i>On Off</i></a>	On
Inherit ProxyPass directives defined from the main server	
<a href="#">ProxyPassInterpolateEnv <i>On Off</i></a>	Off
Enable Environment Variable interpolation in Reverse Proxy configurations	
<a href="#">ProxyPassMatch <i>[regex] ! url [key=value [key=value ...]]</i></a>	
Maps remote servers into the local server URL-space using regular expressions	
<a href="#">ProxyPassReverse <i>[path] url [interpolate]</i></a>	
Adjusts the URL in HTTP response headers sent from a reverse proxied server	
<a href="#">ProxyPassReverseCookieDomain <i>internal-domain public-domain [interpolate]</i></a>	
Adjusts the Domain string in Set-Cookie headers from a reverse- proxied server	
<a href="#">ProxyPassReverseCookiePath <i>internal-path</i></a>	

<a href="#"><i>public-path [interpolate]</i></a>	
Adjusts the Path string in Set-Cookie headers from a reverse-proxied server	
<a href="#">ProxyPreserveHost On Off</a>	Off
Use incoming Host HTTP request header for proxy request	
<a href="#">ProxyReceiveBufferSize bytes</a>	0
Network buffer size for proxied HTTP and FTP connections	
<a href="#">ProxyRemote match remote-server</a>	
Remote proxy used to handle certain requests	
<a href="#">ProxyRemoteMatch regex remote-server</a>	
Remote proxy used to handle requests matched by regular expressions	
<a href="#">ProxyRequests On Off</a>	Off
Enables forward (standard) proxy requests	
<a href="#">ProxySCGIInternalRedirect On Off Headername</a>	On
Enable or disable internal redirect responses from the backend	
<a href="#">ProxySCGISendfile On Off Headername</a>	Off
Enable evaluation of X-Sendfile pseudo response header	
<a href="#">ProxySet url key=value [key=value ...]</a>	
Set various Proxy balancer or member parameters	
<a href="#">ProxySourceAddress address</a>	
Set local IP address for outgoing proxy connections	
<a href="#">ProxyStatus Off On Full</a>	Off
Show Proxy LoadBalancer status in mod_status	
<a href="#">ProxyTimeout seconds</a>	
Network timeout for proxied requests	
<a href="#">ProxyVia On Off Full Block</a>	Off
Information provided in the Via HTTP response header for proxied requests	
<a href="#">QualifyRedirectURL ON OFF</a>	OFF
Controls whether the REDIRECT_URL environment variable is fully qualified	
<a href="#">ReadmeName filename</a>	
<a href="#">ReceiveBufferSize bytes</a>	0
TCP receive buffer size	
<a href="#">Redirect [status] URL-path URL</a>	
URL	

<a href="#">RedirectMatch</a> <i>[status] regex URL</i>	
URL	
<a href="#">RedirectPermanent</a> <i>URL-path URL</i>	
URL	
<a href="#">RedirectTemp</a> <i>URL-path URL</i>	
URL	
<a href="#">ReflectorHeader</a> <i>inputheader [outputheader]</i>	
Reflect an input header to the output headers	
<a href="#">RegisterHttpMethod</a> <i>method [method [...]]</i>	
Register non-standard HTTP methods	
<a href="#">RemoteIPHeader</a> <i>header-field</i>	
Declare the header field which should be parsed for useragent IP addresses	
<a href="#">RemoteIPInternalProxy</a> <i>proxy-ip proxy-ip/subnet hostname ...</i>	
Declare client intranet IP addresses trusted to present the RemoteIPHeader value	
<a href="#">RemoteIPInternalProxyList</a> <i>filename</i>	
Declare client intranet IP addresses trusted to present the RemoteIPHeader value	
<a href="#">RemoteIPProxiesHeader</a> <i>HeaderFieldName</i>	
Declare the header field which will record all intermediate IP addresses	
<a href="#">RemoteIPTrustedProxy</a> <i>proxy-ip proxy-ip/subnet hostname ...</i>	
Declare client intranet IP addresses trusted to present the RemoteIPHeader value	
<a href="#">RemoteIPTrustedProxyList</a> <i>filename</i>	
Declare client intranet IP addresses trusted to present the RemoteIPHeader value	
<a href="#">RemoveCharset</a> <i>extension [extension] ...</i>	
Removes any character set associations for a set of file extensions	
<a href="#">RemoveEncoding</a> <i>extension [extension] ...</i>	
Removes any content encoding associations for a set of file extensions	
<a href="#">RemoveHandler</a> <i>extension [extension] ...</i>	
Removes any handler associations for a set of file extensions	
<a href="#">RemoveInputFilter</a> <i>extension [extension] ...</i>	
Removes any input filter associations for a set of file extensions	
<a href="#">RemoveLanguage</a> <i>extension [extension] ...</i>	
Removes any language associations for a set of file extensions	
<a href="#">RemoveOutputFilter</a> <i>extension [extension] ...</i>	

<a href="#">RemoveType <i>extension</i> [<i>extension</i>] ...</a>	
Removes any output filter associations for a set of file extensions	
<a href="#">RemoveType <i>extension</i> [<i>extension</i>] ...</a>	
Removes any content type associations for a set of file extensions	
<a href="#">RequestHeader <i>set append add unset header</i> [<i>value</i>] [<i>early env=</i>!<i>variable</i>]</a>	
HTTP	
<a href="#">RequestReadTimeout [<i>header=timeout</i>[-<i>maxtimeout</i>][,<i>MinRate=rate</i>] [<i>body=timeout</i>[-<i>maxtimeout</i>][,<i>MinRate=rate</i>]</a>	
Set timeout values for receiving request headers and body from client.	
<a href="#">Require [<i>not</i>] <i>entity-name</i> [<i>entity-name</i>] ...</a>	
Tests whether an authenticated user is authorized by an authorization provider.	
<a href="#">&lt;RequireAll&gt; ... &lt;/RequireAll&gt;</a>	
Enclose a group of authorization directives of which none must fail and at least one must succeed for the enclosing directive to succeed.	
<a href="#">&lt;RequireAny&gt; ... &lt;/RequireAny&gt;</a>	
Enclose a group of authorization directives of which one must succeed for the enclosing directive to succeed.	
<a href="#">&lt;RequireNone&gt; ... &lt;/RequireNone&gt;</a>	
Enclose a group of authorization directives of which none must succeed for the enclosing directive to succeed.	
<a href="#">RewriteBase <i>URL-path</i></a>	
Sets the base URL for per-directory rewrites	
<a href="#">RewriteCond <i>TestString CondPattern</i> [<i>flags</i>]</a>	
Defines a condition under which rewriting will take place	
<a href="#">RewriteEngine <i>on off</i></a>	off
Enables or disables runtime rewriting engine	
<a href="#">RewriteMap <i>MapName MapType:MapSource</i></a>	
Defines a mapping function for key-lookup	
<a href="#">RewriteOptions <i>Options</i></a>	
Sets some special options for the rewrite engine	
<a href="#">RewriteRule <i>Pattern Substitution</i> [<i>flags</i>]</a>	
Defines rules for the rewriting engine	
<a href="#">RLimitCPU <i>seconds max</i> [<i>seconds max</i>]</a>	
Limits the CPU consumption of processes launched by Apache httpd children	
<a href="#">RLimitMEM <i>bytes max</i> [<i>bytes max</i>]</a>	
Limits the memory consumption of processes launched by Apache httpd children	

<a href="#">RLimitNPROC <i>number max [number max]</i></a>	
Limits the number of processes that can be launched by processes launched by Apache http	
<a href="#">Satisfy Any All</a>	All
Interaction between host-level access control and user authentication	
<a href="#">ScoreBoardFile <i>file-path</i></a>	logs/apache_runtime
Location of the file used to store coordination data for the child processes	
<a href="#">Script <i>method cgi-script</i></a>	
CGI	
<a href="#">ScriptAlias <i>URL-path file-path directory-path</i></a>	
URL CGI	
<a href="#">ScriptAliasMatch <i>regex file-path directory-path</i></a>	
URL CGI	
<a href="#">ScriptInterpreterSource Registry Registry-Strict Script</a>	Script
Technique for locating the interpreter for CGI scripts	
<a href="#">ScriptLog <i>file-path</i></a>	
CGI	
<a href="#">ScriptLogBuffer <i>bytes</i></a>	1024
PUT POST	
<a href="#">ScriptLogLength <i>bytes</i></a>	10385760
CGI	
<a href="#">ScriptSock <i>file-path</i></a>	logs/cgisock
cgi	
<a href="#">SecureListen [<i>IP-address:]portnumber Certificate-Name [MUTUAL]</i></a>	
Enables SSL encryption for the specified port	
<a href="#">SeeRequestTail On Off</a>	Off
Determine if mod_status displays the first 63 characters of a request or the last 63, assuming is greater than 63 chars.	
<a href="#">SendBufferSize <i>bytes</i></a>	0
TCP buffer size	
<a href="#">ServerAdmin <i>email-address URL</i></a>	
Email address that the server includes in error messages sent to the client	
<a href="#">ServerAlias <i>hostname [hostname] ...</i></a>	
Alternate names for a host used when matching requests to name-virtual hosts	

<a href="#"><u>ServerLimit</u></a> <i>number</i>	
Upper limit on configurable number of processes	
<a href="#"><u>ServerName</u></a> [ <i>scheme://domain-name</i> ] <a href="#"><u>ip-address</u></a> [: <i>port</i> ]	
Hostname and port that the server uses to identify itself	
<a href="#"><u>ServerPath</u></a> <i>URL-path</i>	
Legacy URL pathname for a name-based virtual host that is accessed by an incompatible b	
<a href="#"><u>ServerRoot</u></a> <i>directory-path</i>	/usr/local/apache
Base directory for the server installation	
<a href="#"><u>ServerSignature</u></a> On Off EMail	Off
Configures the footer on server-generated documents	
<a href="#"><u>ServerTokens</u></a>	Full
<a href="#"><u>Major</u></a>   <a href="#"><u>Minor</u></a>   <a href="#"><u>Min</u></a> [ <i>imal</i> ]  <a href="#"><u>Prod</u></a> [ <i>uctOnly</i> ]  <a href="#"><u>OS</u></a>   <a href="#"><u>Full</u></a>	
Configures the Server HTTP response header	
<a href="#"><u>Session</u></a> On Off	Off
Enables a session for the current directory or location	
<a href="#"><u>SessionCookieName</u></a> <i>name</i> <i>attributes</i>	
Name and attributes for the RFC2109 cookie storing the session	
<a href="#"><u>SessionCookieName2</u></a> <i>name</i> <i>attributes</i>	
Name and attributes for the RFC2965 cookie storing the session	
<a href="#"><u>SessionCookieRemove</u></a> On Off	Off
Control for whether session cookies should be removed from incoming HTTP headers	
<a href="#"><u>SessionCryptoCipher</u></a> <i>name</i>	
The crypto cipher to be used to encrypt the session	
<a href="#"><u>SessionCryptoDriver</u></a> <i>name</i> [ <i>param</i> [= <i>value</i> ]]	
The crypto driver to be used to encrypt the session	
<a href="#"><u>SessionCryptoPassphrase</u></a> <i>secret</i> [ <i>secret</i> ... ]	
The key used to encrypt the session	
<a href="#"><u>SessionCryptoPassphraseFile</u></a> <i>filename</i>	
File containing keys used to encrypt the session	
<a href="#"><u>SessionDBDCookieName</u></a> <i>name</i> <i>attributes</i>	
Name and attributes for the RFC2109 cookie storing the session ID	
<a href="#"><u>SessionDBDCookieName2</u></a> <i>name</i> <i>attributes</i>	
Name and attributes for the RFC2965 cookie storing the session ID	
<a href="#"><u>SessionDBDCookieRemove</u></a> On Off	On

	Control for whether session ID cookies should be removed from incoming HTTP headers
<a href="#"><u>SessionDBDeleteLabel <i>label</i></u></a>	deletesession
	The SQL query to use to remove sessions from the database
<a href="#"><u>SessionDBInsertLabel <i>label</i></u></a>	insertsession
	The SQL query to use to insert sessions into the database
<a href="#"><u>SessionDBDPerUser On Off</u></a>	Off
	Enable a per user session
<a href="#"><u>SessionDBDSelectLabel <i>label</i></u></a>	selectsession
	The SQL query to use to select sessions from the database
<a href="#"><u>SessionDBDUpdateLabel <i>label</i></u></a>	updatesession
	The SQL query to use to update existing sessions in the database
<a href="#"><u>SessionEnv On Off</u></a>	Off
	Control whether the contents of the session are written to the <code>HTTP_SESSION</code> environmen
<a href="#"><u>SessionExclude <i>path</i></u></a>	
	Define URL prefixes for which a session is ignored
<a href="#"><u>SessionHeader <i>header</i></u></a>	
	Import session updates from a given HTTP response header
<a href="#"><u>SessionInclude <i>path</i></u></a>	
	Define URL prefixes for which a session is valid
<a href="#"><u>SessionMaxAge <i>maxage</i></u></a>	0
	Define a maximum age in seconds for a session
<a href="#"><u>SetEnv <i>env-variable value</i></u></a>	
<a href="#"><u>SetEnvIf <i>attribute regex [!]env-variable[=value] [!]env-variable[=value]] ...</i></u></a>	
	Sets environment variables based on an <code>ap_expr</code> expression
<a href="#"><u>SetEnvIfNoCase <i>attribute regex [!]env-variable[=value] [!]env-variable[=value]] ...</i></u></a>	
<a href="#"><u>SetHandler <i>handler-name none expression</i></u></a>	
	Forces all matching files to be processed by a handler
<a href="#"><u>SetInputFilter <i>filter[:filter...]</i></u></a>	
	Sets the filters that will process client requests and POST input

<a href="#">SetOutputFilter <i>filter[:filter...]</i></a>	
Sets the filters that will process responses from the server	
<a href="#">SSIEndTag <i>tag</i></a>	"-->"
String that ends an include element	
<a href="#">SSLErrorMsg <i>message</i></a>	"[an error occurred +
Error message displayed when there is an SSI error	
<a href="#">SSIETag <i>on off</i></a>	off
Controls whether ETags are generated by the server.	
<a href="#">SSILastModified <i>on off</i></a>	off
Controls whether Last -Modified headers are generated by the server.	
<a href="#">SSILegacyExprParser <i>on off</i></a>	off
Enable compatibility mode for conditional expressions.	
<a href="#">SSIStartTag <i>tag</i></a>	"<!--#"
String that starts an include element	
<a href="#">SSITimeFormat <i>formatstring</i></a>	"%A, %d-%b-%Y %H: +
Configures the format in which date strings are displayed	
<a href="#">SSIUndefinedEcho <i>string</i></a>	"(none)"
String displayed when an unset variable is echoed	
<a href="#">SSLCACertificateFile <i>file-path</i></a>	
File of concatenated PEM-encoded CA Certificates for Client Auth	
<a href="#">SSLCACertificatePath <i>directory-path</i></a>	
Directory of PEM-encoded CA Certificates for Client Auth	
<a href="#">SSLCADNRequestFile <i>file-path</i></a>	
File of concatenated PEM-encoded CA Certificates for defining acceptable CA names	
<a href="#">SSLCADNRequestPath <i>directory-path</i></a>	
Directory of PEM-encoded CA Certificates for defining acceptable CA names	
<a href="#">SSLCARevocationCheck <i>chain leaf none flags</i></a>	none
Enable CRL-based revocation checking	
<a href="#">SSLCARevocationFile <i>file-path</i></a>	
File of concatenated PEM-encoded CA CRLs for Client Auth	
<a href="#">SSLCARevocationPath <i>directory-path</i></a>	
Directory of PEM-encoded CA CRLs for Client Auth	
<a href="#">SSLCertificateChainFile <i>file-path</i></a>	
File of PEM-encoded Server CA Certificates	

<a href="#"><u>SSLCertificateFile</u></a> <i>file-path</i>	
Server PEM-encoded X.509 certificate data file	
<a href="#"><u>SSLCertificateKeyFile</u></a> <i>file-path</i>	
Server PEM-encoded private key file	
<a href="#"><u>SSLCipherSuite</u></a> <i>cipher-spec</i>	DEFAULT (depends o
Cipher Suite available for negotiation in SSL handshake	
<a href="#"><u>SSLCompression</u></a> on off	off
Enable compression on the SSL level	
<a href="#"><u>SSLCryptoDevice</u></a> <i>engine</i>	builtin
Enable use of a cryptographic hardware accelerator	
<a href="#"><u>SSLEngine</u></a> on off optional	off
SSL Engine Operation Switch	
<a href="#"><u>SSLFIPS</u></a> on off	off
SSL FIPS mode Switch	
<a href="#"><u>SSLHonorCipherOrder</u></a> on off	off
Option to prefer the server's cipher preference order	
<a href="#"><u>SSLInsecureRenegotiation</u></a> on off	off
Option to enable support for insecure renegotiation	
<a href="#"><u>SSLOCSDefaultResponder</u></a> <i>uri</i>	
Set the default responder URI for OCSP validation	
<a href="#"><u>SSLOCSPEnable</u></a> on off	off
Enable OCSP validation of the client certificate chain	
<a href="#"><u>SSLOCSPNoverify</u></a> On/Off	Off
skip the OCSP responder certificates verification	
<a href="#"><u>SSLOCSPOverrideResponder</u></a> on off	off
Force use of the default responder URI for OCSP validation	
<a href="#"><u>SSLOCSPProxyURL</u></a> <i>url</i>	
Proxy URL to use for OCSP requests	
<a href="#"><u>SSLOCSResponderCertificateFile</u></a> <i>file</i>	
Set of trusted PEM encoded OCSP responder certificates	
<a href="#"><u>SSLOCSResponderTimeout</u></a> <i>seconds</i>	10
Timeout for OCSP queries	
<a href="#"><u>SSLOCSResponseMaxAge</u></a> <i>seconds</i>	-1
Maximum allowable age for OCSP responses	
<a href="#"><u>SSLOCSResponseTimeSkew</u></a> <i>seconds</i>	300

Maximum allowable time skew for OCSP response validation	
<a href="#">SSL_OCSPUseRequestNonce</a> <b>on off</b>	on
Use a nonce within OCSP queries	
<a href="#">SSL_OpenSSLConfCmd</a> <i>command-name</i> <i>command-value</i>	
Configure OpenSSL parameters through its <i>SSL_CONF</i> API	
<a href="#">SSLOptions</a> <b>[+ -]option ...</b>	
Configure various SSL engine run-time options	
<a href="#">SSLPassPhraseDialog</a> <i>type</i>	builtin
Type of pass phrase dialog for encrypted private keys	
<a href="#">SSLProtocol</a> <b>[+ -]protocol ...</b>	all -SSLv3 (up to 2 +
Configure usable SSL/TLS protocol versions	
<a href="#">SSLProxyCACertificateFile</a> <i>file-path</i>	
File of concatenated PEM-encoded CA Certificates for Remote Server Auth	
<a href="#">SSLProxyCACertificatePath</a> <i>directory-path</i>	
Directory of PEM-encoded CA Certificates for Remote Server Auth	
<a href="#">SSLProxyCARevocationCheck</a> <a href="#">chain leaf none</a>	none
Enable CRL-based revocation checking for Remote Server Auth	
<a href="#">SSLProxyCARevocationFile</a> <i>file-path</i>	
File of concatenated PEM-encoded CA CRLs for Remote Server Auth	
<a href="#">SSLProxyCARevocationPath</a> <i>directory-path</i>	
Directory of PEM-encoded CA CRLs for Remote Server Auth	
<a href="#">SSLProxyCheckPeerCN</a> <b>on off</b>	on
Whether to check the remote server certificate's CN field	
<a href="#">SSLProxyCheckPeerExpire</a> <b>on off</b>	on
Whether to check if remote server certificate is expired	
<a href="#">SSLProxyCheckPeerName</a> <b>on off</b>	on
Configure host name checking for remote server certificates	
<a href="#">SSLProxyCipherSuite</a> <i>cipher-spec</i>	ALL:!ADH:RC4+RSA:
Cipher Suite available for negotiation in SSL proxy handshake	
<a href="#">SSLProxyEngine</a> <b>on off</b>	off
SSL Proxy Engine Operation Switch	
<a href="#">SSLProxyMachineCertificateChainFile</a> <i>filename</i>	

<a href="#"><u>SSLProxyMachineCertificateFile <i>filename</i></u></a>	
File of concatenated PEM-encoded CA certificates to be used by the proxy for choosing a c	
<a href="#"><u>SSLProxyMachineCertificatePath <i>directory</i></u></a>	
Directory of PEM-encoded client certificates and keys to be used by the proxy	
<a href="#"><u>SSLProxyProtocol [+ -]<i>protocol</i> ...</u></a>	all -SSLv3 (up to 2 +
Configure usable SSL protocol flavors for proxy usage	
<a href="#"><u>SSLProxyVerify <i>level</i></u></a>	none
Type of remote server Certificate verification	
<a href="#"><u>SSLProxyVerifyDepth <i>number</i></u></a>	1
Maximum depth of CA Certificates in Remote Server Certificate verification	
<a href="#"><u>SSLRandomSeed <i>context source [bytes]</i></u></a>	
Pseudo Random Number Generator (PRNG) seeding source	
<a href="#"><u>SSLRenegBufferSize <i>bytes</i></u></a>	131072
Set the size for the SSL renegotiation buffer	
<a href="#"><u>SSLRequire <i>expression</i></u></a>	
Allow access only when an arbitrarily complex boolean expression is true	
<a href="#"><u>SSLRequireSSL</u></a>	
Deny access when SSL is not used for the HTTP request	
<a href="#"><u>SSLSessionCache <i>type</i></u></a>	none
Type of the global/inter-process SSL Session Cache	
<a href="#"><u>SSLSessionCacheTimeout <i>seconds</i></u></a>	300
Number of seconds before an SSL session expires in the Session Cache	
<a href="#"><u>SSLSessionTicketKeyFile <i>file-path</i></u></a>	
Persistent encryption/decryption key for TLS session tickets	
<a href="#"><u>SSLSessionTickets <i>on off</i></u></a>	on
Enable or disable use of TLS session tickets	
<a href="#"><u>SSLSRPUnknownUserSeed <i>secret-string</i></u></a>	
SRP unknown user seed	
<a href="#"><u>SSLSRPVerifierFile <i>file-path</i></u></a>	
Path to SRP verifier file	
<a href="#"><u>SSLStaplingCache <i>type</i></u></a>	
Configures the OCSP stapling cache	
<a href="#"><u>SSLStaplingErrorCacheTimeout <i>seconds</i></u></a>	600
Number of seconds before expiring invalid responses in the OCSP stapling cache	

<a href="#">SSLStaplingFakeTryLater</a> <i>on off</i>	on
Synthesize "tryLater" responses for failed OCSP stapling queries	
<a href="#">SSLStaplingForceURL</a> <i>uri</i>	
Override the OCSP responder URI specified in the certificate's AIA extension	
<a href="#">SSLStaplingResponderTimeout</a> <i>seconds</i>	10
Timeout for OCSP stapling queries	
<a href="#">SSLStaplingResponseMaxAge</a> <i>seconds</i>	-1
Maximum allowable age for OCSP stapling responses	
<a href="#">SSLStaplingResponseTimeSkew</a> <i>seconds</i>	300
Maximum allowable time skew for OCSP stapling response validation	
<a href="#">SSLStaplingReturnResponderErrors</a> <i>on off</i>	on
Pass stapling related OCSP errors on to client	
<a href="#">SSLStaplingStandardCacheTimeout</a> <i>seconds</i>	3600
Number of seconds before expiring responses in the OCSP stapling cache	
<a href="#">SSLStrictSNIVHostCheck</a> <i>on off</i>	off
Whether to allow non-SNI clients to access a name-based virtual host.	
<a href="#">SSLUserName</a> <i>varname</i>	
Variable name to determine user name	
<a href="#">SSLUseStapling</a> <i>on off</i>	off
Enable stapling of OCSP responses in the TLS handshake	
<a href="#">SSLVerifyClient</a> <i>level</i>	none
Type of Client Certificate verification	
<a href="#">SSLVerifyDepth</a> <i>number</i>	1
Maximum depth of CA Certificates in Client Certificate verification	
<a href="#">StartServers</a> <i>number</i>	
Number of child server processes created at startup	
<a href="#">StartThreads</a> <i>number</i>	
Number of threads created on startup	
<a href="#">Substitute</a> <i>s/pattern/substitution/[infq]</i>	
Pattern to filter the response content	
<a href="#">SubstituteInheritBefore</a> <i>on off</i>	off
Change the merge order of inherited patterns	
<a href="#">SubstituteMaxLineLength</a> <i>bytes(b B k K m M g G)</i>	1m
Set the maximum line size	

<a href="#">Suexec On Off</a>	
Enable or disable the suEXEC feature	
<a href="#">SuexecUserGroup <i>User Group</i></a>	
CGI	
<a href="#">ThreadLimit <i>number</i></a>	
Sets the upper limit on the configurable number of threads per child process	
<a href="#">ThreadsPerChild <i>number</i></a>	
Number of threads created by each child process	
<a href="#">ThreadStackSize <i>size</i></a>	
The size in bytes of the stack used by threads handling client connections	
<a href="#">TimeOut <i>seconds</i></a>	60
Amount of time the server will wait for certain events before failing a request	
<a href="#">TraceEnable [<i>on off extended</i>]</a>	on
Determines the behavior on TRACE requests	
<a href="#">TransferLog <i>file pipe</i></a>	
<a href="#">TypesConfig <i>file-path</i></a>	conf/mime.types
The location of the mime.types file	
<a href="#">UnDefine <i>parameter-name</i></a>	
Undefine the existence of a variable	
<a href="#">UndefMacro <i>name</i></a>	
Undefine a macro	
<a href="#">UnsetEnv <i>env-variable [env-variable] ...</i></a>	
<a href="#">Use <i>name [value1 ... valueN]</i></a>	
Use a macro	
<a href="#">UseCanonicalName On Off DNS</a>	Off
Configures how the server determines its own name and port	
<a href="#">UseCanonicalPhysicalPort On Off</a>	Off
Configures how the server determines its own port	
<a href="#">User <i>unix-userid</i></a>	#-1
The userid under which the server will answer requests	
<a href="#">UserDir <i>directory-filename</i></a>	public_html
<a href="#">VHostCGIMode On Off Secure</a>	On

<a href="#">VHostPrivs</a> <i>[+]?privilege-name</i> <i>[+]?privilege-name</i> ...	
Assign arbitrary privileges to subprocesses created by a virtual host.	
<a href="#">VHostGroup</a> <i>unix-groupid</i>	
Sets the Group ID under which a virtual host runs.	
<a href="#">VHostPrivs</a> <i>[+]?privilege-name</i> <i>[+]?privilege-name</i> ...	
Assign arbitrary privileges to a virtual host.	
<a href="#">VHostSecure</a> On Off	On
Determines whether the server runs with enhanced security for the virtualhost.	
<a href="#">VHostUser</a> <i>unix-userid</i>	
Sets the User ID under which a virtual host runs.	
<a href="#">VirtualDocumentRoot</a> <i>interpolated-directory</i>  none	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">VirtualDocumentRootIP</a> <i>interpolated-directory</i>  none	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">&lt;VirtualHost</a> <i>addr[:port]</i> <i>[addr[:port]]</i> ...> ... <a href="#">&lt;/VirtualHost&gt;</a>	
Contains directives that apply only to a specific hostname or IP address	
<a href="#">VirtualScriptAlias</a> <i>interpolated-directory</i>  none	none
Dynamically configure the location of the CGI directory for a given virtual host	
<a href="#">VirtualScriptAliasIP</a> <i>interpolated-directory</i>  none	none
Dynamically configure the location of the CGI directory for a given virtual host	
<a href="#">WatchdogInterval</a> <i>number-of-seconds</i>	1
Watchdog interval in seconds	
<a href="#">XBitHack</a> on off full	off
Parse SSI directives in files with the execute bit set	
<a href="#">xml2EncAlias</a> <i>charset alias</i> <i>[alias ...]</i>	
Recognise Aliases for encoding values	
<a href="#">xml2EncDefault</a> <i>name</i>	
Sets a default encoding to assume when absolutely no information can be <a href="#">automatically de</a>	

## [xml2StartParse \*element \[element ...\]\*](#)

Advise the parser to skip leading junk.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



.

.

.



.

(MPM)



---

## core

Core Apache HTTP Server features that are always available

## mpm\_common

A collection of directives that are implemented by more than one multi-processing module (MPM)

## event

A variant of the worker MPM with the goal of consuming threads only for connections with active processing

## mpm\_network

Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare

## mpmt\_os2

Hybrid multi-process, multi-threaded MPM for OS/2

## prefork

Implements a non-threaded, pre-forking web server

## mpm\_winnt

Multi-Processing Module optimized for Windows NT.

## worker

Multi-Processing Module implementing a hybrid multi-threaded multi-process web server



A | B | C | D | E | F | H | I | L | M | N | P | R | S |  
U | V | W | X

### [mod\\_access\\_compat](#)

Group authorizations based on host (name or IP address)

### [mod\\_actions](#)

CGI .

### [mod\\_alias](#)

,

URL

### [mod\\_allowmethods](#)

Easily restrict what HTTP methods can be used on the server

### [mod\\_asis](#)

HTTP

### [mod\\_auth\\_basic](#)

Basic authentication

### [mod\\_auth\\_digest](#)

MD5 Digest Authentication .

### [mod\\_auth\\_form](#)

Form authentication

### [mod\\_authn\\_anon](#)

"(anonymous)"

### [mod\\_authn\\_core](#)

Core Authentication

### [mod\\_authn\\_dbd](#)

User authentication using an SQL database

### [mod\\_authn\\_dbm](#)

DBM

### [mod\\_authn\\_file](#)

### [mod\\_auth\\_socache](#)

Manages a cache of authentication credentials to relieve the load on backends

### [mod\\_authnz\\_fcgi](#)

Allows a FastCGI authorizer application to handle Apache httpd authentication and authorization

### [mod\\_authnz\\_ldap](#)

Allows an LDAP directory to be used to store the database for HTTP Basic authentication.

### [mod\\_authz\\_core](#)

Core Authorization

### [mod\\_authz\\_dbd](#)

Group Authorization and Login using SQL

### [mod\\_authz\\_dbm](#)

DBM

### [mod\\_authz\\_groupfile](#)

### [mod\\_authz\\_host](#)

Group authorizations based on host (name or IP address)

### [mod\\_authz\\_owner](#)

### [mod\\_authz\\_user](#)

### [mod\\_autoindex](#)

ls Win32 dir

### [mod\\_brotli](#)

Compress content via Brotli before it is delivered to the client

### [mod\\_buffer](#)

Support for request buffering

### [mod\\_cache](#)

URI .

### [mod\\_cache\\_disk](#)

Content cache storage manager keyed to URIs

### [mod\\_cache\\_socache](#)

Shared object cache (socache) based storage module for the HTTP caching filter.

### [mod\\_cern\\_meta](#)

CERN

### [mod\\_cgi](#)

CGI

### [mod\\_cgid](#)

CGI CGI

### [mod\\_charset\\_lite](#)

### [mod\\_data](#)

Convert response body into an RFC2397 data URL

### [mod\\_dav](#)

Distributed Authoring and Versioning ([WebDAV](#))

### [mod\\_dav\\_fs](#)

[mod\\_dav](#)

### [mod\\_dav\\_lock](#)

Generic locking module for [mod\\_dav](#)

### [mod\\_dbd](#)

Manages SQL database connections

### [mod\\_deflate](#)

### [mod\\_dialup](#)

Send static content at a bandwidth rate limit, defined by the

various old modem standards

### [mod\\_dir](#)

" " index

### [mod\\_dumpio](#)

Dumps all I/O to error log as desired.

### [mod\\_echo](#)

echo

### [mod\\_env](#)

CGI SSI

### [mod\\_example\\_hooks](#)

API

### [mod\\_expires](#)

Expires Cache-Control HTTP

### [mod\\_ext\\_filter](#)

### [mod\\_file\\_cache](#)

### [mod\\_filter](#)

Context-sensitive smart filter configuration module

### [mod\\_headers](#)

HTTP

### [mod\\_heartbeat](#)

Sends messages with server status to frontend proxy

### [mod\\_heartmonitor](#)

Centralized monitor for mod\_heartbeat origin servers

### [mod\\_http2](#)

Support for the HTTP/2 transport layer

### [mod\\_ident](#)

RFC 1413 ident

### [mod\\_imagemap](#)

(imagemap)

### [mod\\_include](#)

Server-parsed html documents (Server Side Includes)

### [mod\\_info](#)

### [mod\\_isapi](#)

Windows ISAPI Extension

### [mod\\_lbmethod\\_bybusyness](#)

Pending Request Counting load balancer scheduler algorithm  
for [mod\\_proxy\\_balancer](#)

### [mod\\_lbmethod\\_byrequests](#)

Request Counting load balancer scheduler algorithm for  
[mod\\_proxy\\_balancer](#)

### [mod\\_lbmethod\\_bytraffic](#)

Weighted Traffic Counting load balancer scheduler algorithm  
for [mod\\_proxy\\_balancer](#)

### [mod\\_lbmethod\\_heartbeat](#)

Heartbeat Traffic Counting load balancer scheduler algorithm  
for [mod\\_proxy\\_balancer](#)

### [mod\\_ldap](#)

LDAP connection pooling and result caching services for use  
by other LDAP modules

### [mod\\_log\\_config](#)

### [mod\\_log\\_debug](#)

Additional configurable debug logging

### [mod\\_log\\_forensic](#)

Forensic Logging of the requests made to the server

## [mod\\_logio](#)

## [mod\\_lua](#)

Provides Lua hooks into various portions of the httpd request processing

## [mod\\_macro](#)

Provides macros within apache httpd runtime configuration files

## [mod\\_mime](#)

Associates the requested filename's extensions with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding)

## [mod\\_mime\\_magic](#)

Determines the MIME type of a file by looking at a few bytes of its contents

## [mod\\_negotiation](#)

Provides for [content negotiation](#)

## [mod\\_nw\\_ssl](#)

Enable SSL encryption for NetWare

## [mod\\_privileges](#)

Support for Solaris privileges and for running virtual hosts under different user IDs.

## [mod\\_proxy](#)

Multi-protocol proxy/gateway server

## [mod\\_proxy\\_ajp](#)

AJP support module for [mod\\_proxy](#)

## [mod\\_proxy\\_balancer](#)

[mod\\_proxy](#) extension for load balancing

## [mod\\_proxy\\_connect](#)

[mod\\_proxy](#) extension for CONNECT request handling

### [mod\\_proxy\\_express](#)

Dynamic mass reverse proxy extension for [mod\\_proxy](#)

### [mod\\_proxy\\_fcgi](#)

FastCGI support module for [mod\\_proxy](#)

### [mod\\_proxy\\_fdpass](#)

fdpass external process support module for [mod\\_proxy](#)

### [mod\\_proxy\\_ftp](#)

FTP support module for [mod\\_proxy](#)

### [mod\\_proxy\\_hcheck](#)

Dynamic health check of Balancer members (workers) for [mod\\_proxy](#)

### [mod\\_proxy\\_html](#)

Rewrite HTML links in to ensure they are addressable from Clients' networks in a proxy context.

### [mod\\_proxy\\_http](#)

HTTP support module for [mod\\_proxy](#)

### [mod\\_proxy\\_http2](#)

HTTP/2 support module for [mod\\_proxy](#)

### [mod\\_proxy\\_scgi](#)

SCGI gateway module for [mod\\_proxy](#)

### [mod\\_proxy\\_wstunnel](#)

Websockets support module for [mod\\_proxy](#)

### [mod\\_ratelimit](#)

Bandwidth Rate Limiting for Clients

### [mod\\_reflector](#)

Reflect a request body as a response via the output filter stack.

### [mod\\_remoteip](#)

Replaces the original client IP address for the connection with

the useragent IP address list presented by a proxies or a load balancer via the request headers.

### [mod\\_reqtimeout](#)

Set timeout and minimum data rate for receiving requests

### [mod\\_request](#)

Filters to handle and make available HTTP request bodies

### [mod\\_rewrite](#)

Provides a rule-based rewriting engine to rewrite requested URLs on the fly

### [mod\\_sed](#)

Filter Input (request) and Output (response) content using sed syntax

### [mod\\_session](#)

Session support

### [mod\\_session\\_cookie](#)

Cookie based session support

### [mod\\_session\\_crypto](#)

Session encryption support

### [mod\\_session\\_dbd](#)

DBD/SQL based session support

### [mod\\_setenvif](#)

### [mod\\_slotmem\\_plain](#)

Slot-based shared memory provider.

### [mod\\_slotmem\\_shm](#)

Slot-based shared memory provider.

### [mod\\_so](#)

### [mod\\_socache\\_dbm](#)

DBM based shared object cache provider.

#### [mod\\_socache\\_dc](#)

Distcache based shared object cache provider.

#### [mod\\_socache\\_memcache](#)

Memcache based shared object cache provider.

#### [mod\\_socache\\_shmcb](#)

shmcb based shared object cache provider.

#### [mod\\_speling](#)

URL

#### [mod\\_ssl](#)

Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols

#### [mod\\_status](#)

#### [mod\\_substitute](#)

Perform search and replace operations on response bodies

#### [mod\\_suexec](#)

CGI

#### [mod\\_unique\\_id](#)

#### [mod\\_unixd](#)

Basic (required) security for Unix-family platforms.

#### [mod\\_userdir](#)

#### [mod\\_usertrack](#)

*Clickstream* logging of user activity on a site

#### [mod\\_version](#)

#### [mod\\_vhost\\_alias](#)

Provides for dynamically configured mass virtual hosting

### **mod\_watchdog**

provides infrastructure for other modules to periodically run tasks

### **mod\_xml2enc**

Enhanced charset/internationalisation support for libxml2-based filter modules



| | [FAQ](#) | |



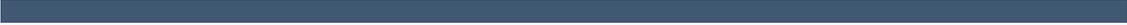
HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



[Apache HTTP Server Version 2.4](#) .



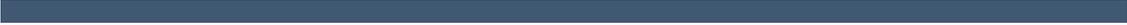


- [1.3 2.0](#)
- [2.0](#)
- 



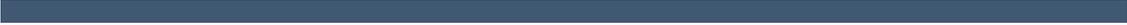
- 
- [\[redacted\]](#)
  - [\[redacted\]](#)
  - [\[redacted\]](#)
  - [\[redacted\]](#)
  - [Directory, Location, Files](#)
  - [\[redacted\]](#)
  - [\[redacted\]](#)
  - [URL](#)
  - [\[redacted\]](#)
  - [\(DSO\)](#)
  - [\(content negotiation\)](#)
  - [\[redacted\]](#)
  - [\[redacted\]](#)
  - [\(MPM\)](#)
  - [\[redacted\]](#)
  - [\[redacted\]](#)
  - [\[redacted\]](#)
  - [suEXEC](#)
  - [\[redacted\]](#)
  - [URL \(rewriting\)](#)





- 
- 
- IP
- 
- 
- 
- (file descriptor)
- DNS





- 
- 
- 



## SSL/TLS

- 
- [SSL/TLS :](#)
- [SSL/TLS :](#)
- [SSL/TLS : How-To](#)
- [SSL/TLS : FAQ](#)



- 
- 
- [CGI](#)
- [Server Side Includes](#)
- [.htaccess](#)
- [\\_\\_\\_\\_\\_](#)





- [Microsoft Windows](#)

- [Microsoft Windows](#)

- [Novell NetWare](#)

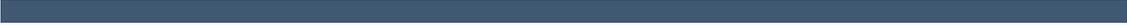
- [HPUX](#)

- [EBCDIC](#)



- 
- - [Manpage: httpd](#)
  - [Manpage: ab](#)
  - [Manpage: apachectl](#)
  - [Manpage: apxs](#)
  - [Manpage: configure](#)
  - [Manpage: dbmmanage](#)
  - [Manpage: htcacheclean](#)
  - [Manpage: htdigest](#)
  - [Manpage: htpasswd](#)
  - [Manpage: logresolve](#)
  - [Manpage: rotatelog](#)
  - [Manpage: suexec](#)
  -





- 
- [\[redacted\]](#)
  - [\[redacted\]](#)
  
  - [\[redacted\]](#)
  - [MPM](#)
  - [MPM event](#)
  - [MPM network](#)
  - [MPM os2](#)
  - [MPM prefork](#)
  - [MPM winnt](#)
  - [MPM worker](#)
  
  - [mod\\_access\\_compat](#)
  - [mod\\_actions](#)
  - [mod\\_alias](#)
  - [mod\\_allowmethods](#)
  - [mod\\_asis](#)
  - [mod\\_auth\\_basic](#)
  - [mod\\_auth\\_digest](#)
  - [mod\\_auth\\_form](#)
  - [mod\\_authn\\_anon](#)
  - [mod\\_authn\\_core](#)
  - [mod\\_authn\\_dbd](#)
  - [mod\\_authn\\_dbm](#)
  - [mod\\_authn\\_file](#)
  - [mod\\_authn\\_socache](#)
  - [mod\\_authnz\\_fcgi](#)
  - [mod\\_authnz\\_ldap](#)
  - [mod\\_authz\\_core](#)
  - [mod\\_authz\\_dbd](#)
  - [mod\\_authz\\_dbm](#)
  - [mod\\_authz\\_groupfile](#)
  - [mod\\_authz\\_host](#)

- [mod\\_authz\\_owner](#)
- [mod\\_authz\\_user](#)
- [mod\\_autoindex](#)
- [mod\\_brotli](#)
- [mod\\_buffer](#)
- [mod\\_cache](#)
- [mod\\_cache\\_disk](#)
- [mod\\_cache\\_socache](#)
- [mod\\_cern\\_meta](#)
- [mod\\_cgi](#)
- [mod\\_cgid](#)
- [mod\\_charset\\_lite](#)
- [mod\\_data](#)
- [mod\\_dav](#)
- [mod\\_dav\\_fs](#)
- [mod\\_dav\\_lock](#)
- [mod\\_dbd](#)
- [mod\\_deflate](#)
- [mod\\_dialup](#)
- [mod\\_dir](#)
- [mod\\_dumpio](#)
- [mod\\_echo](#)
- [mod\\_env](#)
- [mod\\_example\\_hooks](#)
- [mod\\_expires](#)
- [mod\\_ext\\_filter](#)
- [mod\\_file\\_cache](#)
- [mod\\_filter](#)
- [mod\\_headers](#)
- [mod\\_heartbeat](#)
- [mod\\_heartmonitor](#)
- [mod\\_http2](#)
- [mod\\_ident](#)
- [mod\\_imagemap](#)

- [mod\\_include](#)
- [mod\\_info](#)
- [mod\\_isapi](#)
- [mod\\_lbmethod\\_bybusyness](#)
- [mod\\_lbmethod\\_byrequests](#)
- [mod\\_lbmethod\\_bytraffic](#)
- [mod\\_lbmethod\\_heartbeat](#)
- [mod\\_ldap](#)
- [mod\\_log\\_config](#)
- [mod\\_log\\_debug](#)
- [mod\\_log\\_forensic](#)
- [mod\\_logio](#)
- [mod\\_lua](#)
- [mod\\_macro](#)
- [mod\\_mime](#)
- [mod\\_mime\\_magic](#)
- [mod\\_negotiation](#)
- [mod\\_nw\\_ssl](#)
- [mod\\_privileges](#)
- [mod\\_proxy](#)
- [mod\\_proxy\\_ajp](#)
- [mod\\_proxy\\_balancer](#)
- [mod\\_proxy\\_connect](#)
- [mod\\_proxy\\_express](#)
- [mod\\_proxy\\_fcgi](#)
- [mod\\_proxy\\_fdpass](#)
- [mod\\_proxy\\_ftp](#)
- [mod\\_proxy\\_hcheck](#)
- [mod\\_proxy\\_html](#)
- [mod\\_proxy\\_http](#)
- [mod\\_proxy\\_http2](#)
- [mod\\_proxy\\_scgi](#)
- [mod\\_proxy\\_wstunnel](#)
- [mod\\_ratelimit](#)

- [mod\\_reflector](#)
- [mod\\_remoteip](#)
- [mod\\_reqtimeout](#)
- [mod\\_request](#)
- [mod\\_rewrite](#)
- [mod\\_sed](#)
- [mod\\_session](#)
- [mod\\_session\\_cookie](#)
- [mod\\_session\\_crypto](#)
- [mod\\_session\\_dbd](#)
- [mod\\_setenvif](#)
- [mod\\_slotmem\\_plain](#)
- [mod\\_slotmem\\_shm](#)
- [mod\\_so](#)
- [mod\\_socache\\_dbm](#)
- [mod\\_socache\\_dc](#)
- [mod\\_socache\\_memcache](#)
- [mod\\_socache\\_shmcb](#)
- [mod\\_speling](#)
- [mod\\_ssl](#)
- [mod\\_status](#)
- [mod\\_substitute](#)
- [mod\\_suexec](#)
- [mod\\_unique\\_id](#)
- [mod\\_unixd](#)
- [mod\\_userdir](#)
- [mod\\_usertrack](#)
- [mod\\_version](#)
- [mod\\_vhost\\_alias](#)
- [mod\\_watchdog](#)
- [mod\\_xml2enc](#)



---

- 

- [Apache API](#)

- [APR](#)

- [Apache 2.0](#)

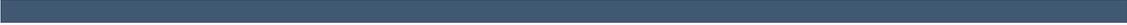
- [Apache 2.0 \(hook\)](#)

- [Apache 1.3 Apache 2.0](#)

- [Apache 2.0](#)

- [Apache 2.0](#)





- 
- 
- 
- 



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

--



.



---

[httpd](#)

[apachectl](#)

[ab](#)

[apxs](#)

(APache eXtenSion tool)

[configure](#)

[dbmmanage](#)

basic authentication DBM

[htcacheclean](#)

[htdigest](#)

digest authentication

[htpasswd](#)

basic authentication

[logresolve](#)

IP-

[rotatelog](#)

[suexec](#)

(Switch User For Exec)

[manpage](#)

manpage .



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## Apache SSL/TLS Encryption

The Apache HTTP Server module `mod_ssl` provides an interface to the [OpenSSL](#) library, which provides Strong Encryption using the Secure Sockets Layer and Transport Layer Security protocols.



- 
- [mod\\_ssl Configuration How-To](#)
  - [Introduction To SSL](#)
  - [Compatibility](#)
  - [Frequently Asked Questions](#)
  - [Glossary](#)



Extensive documentation on the directives and environment variables provided by this module is provided in the [mod\\_ssl reference documentation](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# Apache mod\_rewrite

[mod\\_rewrite](#) provides a way to modify incoming URL requests, dynamically, based on [regular expression](#) rules. This allows you to map arbitrary URLs onto your internal URL structure in any way you like.

It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests: server variables, environment variables, HTTP headers, time stamps, external database lookups, and various other external programs or handlers, can be used to achieve granular URL matching.

Rewrite rules can operate on the full URLs, including the path-info and query string portions, and may be used in per-server context (`httpd.conf`), per-virtualhost context (`<VirtualHost>` blocks), or per-directory context (`.htaccess` files and `<Directory>` blocks). The rewritten result can lead to further rules, internal sub-processing, external request redirection, or proxy passthrough, depending on what [flags](#) you attach to the rules.

Since `mod_rewrite` is so powerful, it can indeed be rather complex. This document supplements the [reference documentation](#), and attempts to allay some of that complexity, and provide highly annotated examples of common scenarios that you may handle with `mod_rewrite`. But we also attempt to show you when you should not use `mod_rewrite`, and use other standard Apache features instead, thus avoiding this unnecessary complexity.

- [mod\\_rewrite reference documentation](#)
- [Introduction to regular expressions and mod\\_rewrite](#)
- [Using mod\\_rewrite for redirection and remapping of URLs](#)

- [Using mod\\_rewrite to control access](#)
- [Dynamic virtual hosts with mod\\_rewrite](#)
- [Dynamic proxying with mod\\_rewrite](#)
- [Using RewriteMap](#)
- [Advanced techniques](#)
- [When \*\*NOT\*\* to use mod\\_rewrite](#)
- [RewriteRule Flags](#)
- [Technical details](#)



## See also

[mod\\_rewrite reference documentation](#)

[Mapping URLs to the Filesystem](#)

[mod\\_rewrite wiki](#)

[Glossary](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



(Virtual Host) ( , www.company1.co  
 www.company2.com) . IP  
 based)" IP " (name-based) " .

IP . 1.1 IP  
 (host-based) IP (non-IP virtual hosts) .

1.3



mod\_vhost\_alias



- 
- (IP )
  - IP ( IP )
  - 
  - (file descriptor)   (, )
  - 
  -



- [<VirtualHost>](#)
- [NameVirtualHost](#)
- [ServerName](#)
- [ServerAlias](#)
- [ServerPath](#)

-S ., :

```
/usr/local/apache2/bin/httpd -S
```

( [httpd](#) .) . IP



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# Frequently Asked Questions

The FAQ has been moved to the [HTTP Server Wiki](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# Developer Documentation for the Apache HTTP Server 2.4

## Warning

Many of the documents listed here are in need of update. They are in different stages of progress. Please be patient and follow [this link](#) to propose a fix or point out any error/discrepancy.



- [Developing modules for the Apache HTTP Server 2.4](#)
- [Hook Functions in 2.4](#)
- [Request Processing in 2.4](#)
- [How filters work in 2.4](#)
- [Guidelines for output filters in 2.4](#)
- [Documenting code in 2.4](#)
- [Thread Safety Issues in 2.4](#)



## Upgrading to 2.x

- [API changes in 2.3/2.4](#)
- [Converting Modules from 1.3 to 2.x](#)



- 
- [Autogenerated Apache HTTP Server \(trunk\) code documentation](#) (the link is built by this [job](#)).
  - Developer articles at [apachetutor](#) include:
    - [Request Processing](#)
    - [Configuration for Modules](#)
    - [Resource Management](#)
    - [Connection Pooling](#)
    - [Introduction to Buckets and Brigades](#)

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)



2.1



(,)  
)



""

."

URL

mod\_rewrite

mod\_rewrite





[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## Overview of new features in Apache HTTP Server 2.4

This document describes some of the major changes between the 2.2 and 2.4 versions of the Apache HTTP Server. For new features since version 2.0, see the [2.2 new features](#) document.



## Run-time Loadable MPMs

Multiple MPMs can now be [built as loadable modules](#) at compile time. The MPM of choice can be configured at run time via [LoadModule](#) directive.

## Event MPM

The [Event MPM](#) is no longer experimental but is now fully supported.

## Asynchronous support

Better support for asynchronous read/write for supporting MPMs and platforms.

## Per-module and per-directory LogLevel configuration

The [LogLevel](#) can now be configured per module and per directory. New levels `trace1` to `trace8` have been added above the debug log level.

## Per-request configuration sections

[<If>](#), [<ElseIf>](#), and [<Else>](#) sections can be used to set the configuration based on per-request criteria.

## General-purpose expression parser

A new expression parser allows to specify [complex conditions](#) using a common syntax in directives like [SetEnvIfExpr](#), [RewriteCond](#), [Header](#), [<If>](#), and others.

## KeepAliveTimeout in milliseconds

It is now possible to specify [KeepAliveTimeout](#) in milliseconds.

## NameVirtualHost directive

No longer needed and is now deprecated.

## Override Configuration

The new [AllowOverrideList](#) directive allows more fine grained control which directives are allowed in `.htaccess`

files.

### **Config file variables**

It is now possible to Define variables in the configuration, allowing a clearer representation if the same value is used at many places in the configuration.

### **Reduced memory usage**

Despite many new features, 2.4.x tends to use less memory than 2.2.x.



### **mod\_proxy\_fcgi**

FastCGI Protocol backend for [mod\\_proxy](#)

### **mod\_proxy\_scgi**

SCGI Protocol backend for [mod\\_proxy](#)

### **mod\_proxy\_express**

Provides dynamically configured mass reverse proxies for [mod\\_proxy](#)

### **mod\_remoteip**

Replaces the apparent client remote IP address and hostname for the request with the IP address list presented by a proxies or a load balancer via the request headers.

### **mod\_heartmonitor, mod\_lbmethod\_heartbeat**

Allow [mod\\_proxy\\_balancer](#) to base loadbalancing decisions on the number of active connections on the backend servers.

### **mod\_proxy\_html**

Formerly a third-party module, this supports fixing of HTML links in a reverse proxy situation, where the backend generates URLs that are not valid for the proxy's clients.

### **mod\_sed**

An advanced replacement of [mod\\_substitute](#), allows to edit the response body with the full power of sed.

### **mod\_auth\_form**

Enables form-based authentication.

### **mod\_session**

Enables the use of session state for clients, using cookie or database storage.

### **mod\_allowmethods**

New module to restrict certain HTTP methods without

interfering with authentication or authorization.

### **mod\_lua**

Embeds the [Lua](#) language into httpd, for configuration and small business logic functions. (Experimental)

### **mod\_log\_debug**

Allows the addition of customizable debug logging at different phases of the request processing.

### **mod\_buffer**

Provides for buffering the input and output filter stacks

### **mod\_data**

Convert response body into an RFC2397 data URL

### **mod\_ratelimit**

Provides Bandwidth Rate Limiting for Clients

### **mod\_request**

Provides Filters to handle and make available HTTP request bodies

### **mod\_reflector**

Provides Reflection of a request body as a response via the output filter stack.

### **mod\_slotmem\_shm**

Provides a Slot-based shared memory provider (ala the scoreboard).

### **mod\_xml2enc**

Formerly a third-party module, this supports internationalisation in libxml2-based (markup-aware) filter modules.

### **mod\_macro (available since 2.4.5)**

Provide macros within configuration files.

### **mod\_proxy\_wstunnel (available since 2.4.5)**

Support web-socket tunnels.

**mod\_authnz\_fcgi (available since 2.4.10)**

Enable FastCGI authorizer applications to authenticate and/or authorize clients.

**mod\_http2 (available since 2.4.17)**

Support for the HTTP/2 transport layer.

**mod\_proxy\_hcheck (available since 2.4.21)**

Support independent dynamic health checks for remote proxy backend servers.



## mod\_ssl

mod\_ssl can now be configured to use an OCSP server to check the validation status of a client certificate. The default responder is configurable, along with the decision on whether to prefer the responder designated in the client certificate itself.

mod\_ssl now also supports OCSP stapling, where the server pro-actively obtains an OCSP verification of its certificate and transmits that to the client during the handshake.

mod\_ssl can now be configured to share SSL Session data between servers through memcached  
EC keys are now supported in addition to RSA and DSA.  
Support for TLS-SRP (available in 2.4.4 and later).

## mod\_proxy

The ProxyPass directive is now most optimally configured within a Location or LocationMatch block, and offers a significant performance advantage over the traditional two-parameter syntax when present in large numbers.

The source address used for proxy requests is now configurable.

Support for Unix domain sockets to the backend (available in 2.4.7 and later).

## mod\_proxy\_balancer

More runtime configuration changes for BalancerMembers via balancer-manager

Additional BalancerMembers can be added at runtime via balancer-manager

Runtime configuration of a subset of Balancer parameters  
BalancerMembers can be set to 'Drain' so that they only respond to existing sticky sessions, allowing them to be taken gracefully offline.

Balancer settings can be persistent after restarts.

## **mod\_cache**

The [mod\\_cache](#) CACHE filter can be optionally inserted at a given point in the filter chain to provide fine control over caching.

[mod\\_cache](#) can now cache HEAD requests.

Where possible, [mod\\_cache](#) directives can now be set per directory, instead of per server.

The base URL of cached URLs can be customised, so that a cluster of caches can share the same endpoint URL prefix.

[mod\\_cache](#) is now capable of serving stale cached data when a backend is unavailable (error 5xx).

[mod\\_cache](#) can now insert HIT/MISS/REVALIDATE into an X-Cache header.

## **mod\_include**

Support for the 'onerror' attribute within an 'include' element, allowing an error document to be served on error instead of the default error string.

## **mod\_cgi, mod\_include, mod\_isapi, ...**

Translation of headers to environment variables is more strict than before to mitigate some possible cross-site-scripting attacks via header injection. Headers containing invalid characters (including underscores) are now silently dropped. [Environment Variables in Apache](#) has some pointers on how to work around broken legacy clients which require such headers. (This affects all modules which use these environment variables.)

## **mod\_auth\_core Authorization Logic Containers**

Advanced authorization logic may now be specified using the [Require](#) directive and the related container directives, such as [<RequireAll>](#).

## **mod\_rewrite**

[mod\\_rewrite](#) adds the [QSD] (Query String Discard) and

[END] flags for [RewriteRule](#) to simplify common rewriting scenarios.

Adds the possibility to use complex boolean expressions in [RewriteCond](#).

Allows the use of SQL queries as [RewriteMap](#) functions.

### **mod\_ldap, mod\_authnz\_ldap**

[mod\\_authnz\\_ldap](#) adds support for nested groups.

[mod\\_ldap](#) adds [LDAPConnectionPoolTTL](#), [LDAPTimeout](#), and other improvements in the handling of timeouts. This is especially useful for setups where a stateful firewall drops idle connections to the LDAP server.

[mod\\_ldap](#) adds [LDAPLibraryDebug](#) to log debug information provided by the used LDAP toolkit.

### **mod\_info**

[mod\\_info](#) can now dump the pre-parsed configuration to stdout during server startup.

### **mod\_auth\_basic**

New generic mechanism to fake basic authentication (available in 2.4.5 and later).



## [fcgistarter](#)

New FastCGI daemon starter utility

## [htcacheclean](#)

Current cached URLs can now be listed, with optional metadata included.

Allow explicit deletion of individual cached URLs from the cache.

File sizes can now be rounded up to the given block size, making the size limits map more closely to the real size on disk.

Cache size can now be limited by the number of inodes, instead of or in addition to being limited by the size of the files on disk.

## [rotatelog](#)

May now create a link to the current log file.

May now invoke a custom post-rotate script.

## [htpasswd](#), [htdbm](#)

Support for the bcrypt algorithm (available in 2.4.4 and later).



## **mod\_rewrite**

The [mod\\_rewrite](#) documentation has been rearranged and almost completely rewritten, with a focus on examples and common usage, as well as on showing you when other solutions are more appropriate. The [Rewrite Guide](#) is now a top-level section with much more detail and better organization.

## **mod\_ssl**

The [mod\\_ssl](#) documentation has been greatly enhanced, with more examples at the getting started level, in addition to the previous focus on technical details.

## **Caching Guide**

The [Caching Guide](#) has been rewritten to properly distinguish between the RFC2616 HTTP/1.1 caching features provided by [mod\\_cache](#), and the generic key/value caching provided by the [socache](#) interface, as well as to cover specialised caching provided by mechanisms such as [mod\\_file\\_cache](#).



## Check Configuration Hook Added

A new hook, `check_config`, has been added which runs between the `pre_config` and `open_logs` hooks. It also runs before the `test_config` hook when the `-t` option is passed to [httpd](#). The `check_config` hook allows modules to review interdependent configuration directive values and adjust them while messages can still be logged to the console. The user can thus be alerted to misconfiguration problems before the core `open_logs` hook function redirects console output to the error log.

## Expression Parser Added

We now have a general-purpose expression parser, whose API is exposed in `ap_expr.h`. This is adapted from the expression parser previously implemented in [mod\\_ssl](#).

## Authorization Logic Containers

Authorization modules now register as a provider, via `ap_register_auth_provider()`, to support advanced authorization logic, such as [<RequireAll>](#).

## Small-Object Caching Interface

The `ap_socache.h` header exposes a provider-based interface for caching small data objects, based on the previous implementation of the [mod\\_ssl](#) session cache. Providers using a shared-memory cyclic buffer, disk-based dbm files, and a memcache distributed cache are currently supported.

## Cache Status Hook Added

The [mod\\_cache](#) module now includes a new `cache_status` hook, which is called when the caching decision becomes known. A default implementation is provided which adds an optional `X-Cache` and `X-Cache-Details` header to the response.

The developer documentation contains a [detailed list of API changes](#).

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## API Changes in Apache HTTP Server 2.4 since 2.2

This document describes changes to the Apache HTTPD API from version 2.2 to 2.4, that may be of interest to module/application developers and core hacks. As of the first GA release of the 2.4 branch API compatibility is preserved for the life of the 2.4 branch. (The [VERSIONING](#) description for the 2.4 release provides more information about API compatibility.)

API changes fall into two categories: APIs that are altogether new, and existing APIs that are expanded or changed. The latter are further divided into those where all changes are backwards-compatible (so existing modules can ignore them), and those that might require attention by maintainers. As with the transition from HTTPD 2.0 to 2.2, existing modules and applications will require recompiling and may call for some attention, but most should not require any substantial updating (although some may be able to take advantage of API changes to offer significant improvements).

For the purpose of this document, the API is split according to the public header files. These headers are themselves the reference documentation, and can be used to generate a browsable HTML reference with `make docs`.



## **ap\_expr (NEW!)**

Introduces a new API to parse and evaluate boolean and algebraic expressions, including provision for a standard syntax and customised variants.

## **ap\_listen (changed; backwards-compatible)**

Introduces a new API to enable httpd child processes to serve different purposes.

## **ap\_mpm (changed)**

ap\_mpm\_run is replaced by a new mpm hook. Also ap\_graceful\_stop\_signalled is lost, and ap\_mpm\_register\_timed\_callback is new.

## **ap\_regex (changed)**

In addition to the existing regexp wrapper, a new higher-level API ap\_rxplus is now provided. This provides the capability to compile Perl-style expressions like `s/regexp/replacement/flags` and to execute them against arbitrary strings. Support for regexp backreferences is also added.

## **ap\_slotmem (NEW!)**

Introduces an API for modules to allocate and manage memory slots, most commonly for shared memory.

## **ap\_socache (NEW!)**

API to manage a shared object cache.

## **heartbeat (NEW!)**

common structures for heartbeat modules

## **ap\_parse\_htaccess (changed)**

The function signature for `ap_parse_htaccess` has been changed. A `apr_table_t` of individual directives allowed for override must now be passed (override remains).

## **http\_config (changed)**

- Introduces per-module, per-directory loglevels, including macro wrappers.
- New `AP_DECLARE_MODULE` macro to declare all modules.
- New `APLOG_USE_MODULE` macro necessary for per-module loglevels in multi-file modules.
- New API to retain data across module unload/load
- New `check_config` hook
- New `ap_process_fnmatch_configs()` function to process wildcards
- Change `ap_configfile_t`, `ap_cfg_getline()`, `ap_cfg_getc()` to return error codes, and add `ap_pcfg_strerror()` for retrieving an error description.
- Any config directive permitted in `ACCESS_CONF` context must now correctly handle being called from an `.htaccess` file via the new [AllowOverrideList](#) directive.  
`ap_check_cmd_context()` accepts a new flag `NOT_IN_HTACCESS` to detect this case.

## **http\_core (changed)**

- REMOVED `ap_default_type`, `ap_requires`, all 2.2 authnz API
- Introduces Optional Functions for logio and authnz
- New function `ap_get_server_name_for_url` to support IPv6 literals.

- New function `ap_register_errorlog_handler` to register error log format string handlers.
- Arguments of `error_log` hook have changed. Declaration has moved to `http_core.h`.
- New function `ap_state_query` to determine if the server is in the initial configuration preflight phase or not. This is both easier to use and more correct than the old method of creating a pool userdata entry in the process pool.
- New function `ap_get_conn_socket` to get the socket descriptor for a connection. This should be used instead of accessing the core connection config directly.

## **httpd (changed)**

- Introduce per-directory, per-module loglevel
- New loglevels `APLOG_TRACE $n$`
- Introduce errorlog ids for requests and connections
- Support for `mod_request` `kept_body`
- Support buffering filter data for async requests
- New `CONN_STATE` values
- Function changes: `ap_escape_html` updated; `ap_unescape_all`, `ap_escape_path_segment_buffer`
- Modules that load other modules later than the `EXEC_ON_READ` config reading stage need to call `ap_reserve_module_slots()` or `ap_reserve_module_slots_directive()` in their `pre_config` hook.
- The useragent IP address per request can now be tracked independently of the client IP address of the connection, for support of deployments with load balancers.

## **http\_log (changed)**

- Introduce per-directory, per-module loglevel

- New loglevels APLOG\_TRACE
- `ap_log_*error` become macro wrappers (backwards-compatible if APLOG\_MARK macro is used, except that is no longer possible to use `#ifdef` inside the argument list)
- piped logging revamped
- `module_index` added to `error_log` hook
- new function: `ap_log_command_line`

### **http\_request (changed)**

- New `auth_internal` API and `auth_provider` API
- New EOR bucket type
- New function `ap_process_async_request`
- New flags `AP_AUTH_INTERNAL_PER_CONF` and `AP_AUTH_INTERNAL_PER_URI`
- New `access_checker_ex` hook to apply additional access control and/or bypass authentication.
- New functions `ap_hook_check_access_ex`, `ap_hook_check_access`, `ap_hook_check_authn`, `ap_hook_check_authz` which accept `AP_AUTH_INTERNAL_PER_*` flags
- DEPRECATED direct use of `ap_hook_access_checker`, `access_checker_ex`, `ap_hook_check_user_id`, `ap_hook_auth_checker`

When possible, registering all access control hooks (including authentication and authorization hooks) using `AP_AUTH_INTERNAL_PER_CONF` is recommended. If all modules' access control hooks are registered with this flag, then whenever the server handles an internal sub-request that matches the same set of access control configuration directives as the initial request (which is the common case), it can avoid invoking the access control hooks another time.

If your module requires the old behavior and must perform access control checks on every sub-request with a different URI from the initial request, even if that URI matches the same set of access control configuration directives, then use `AP_AUTH_INTERNAL_PER_URI`.

### **mod\_auth (NEW!)**

Introduces the new provider framework for authn and authz

### **mod\_cache (changed)**

Introduces a `commit_entity()` function to the cache provider interface, allowing atomic writes to cache. Add a `cache_status()` hook to report the cache decision. All private structures and functions were removed.

### **mod\_core (NEW!)**

This introduces low-level APIs to send arbitrary headers, and exposes functions to handle HTTP OPTIONS and TRACE.

### **mod\_cache\_disk (changed)**

Changes the disk format of the disk cache to support atomic cache updates without locking. The device/inode pair of the body file is embedded in the header file, allowing confirmation that the header and body belong to one another.

### **mod\_disk\_cache (renamed)**

The `mod_disk_cache` module has been renamed to `mod_cache_disk` in order to be consistent with the naming of other modules within the server.

### **mod\_request (NEW!)**

The API for [mod\\_request](#), to make input data available to multiple application/handler modules where required, and to parse HTML form data.

### **mpm\_common (changed)**

- REMOVES: `accept`, `lockfile`, `lock_mech`, `set_scoreboard` (locking uses the new `ap_mutex` API)
- NEW API to drop privileges (delegates this platform-dependent function to modules)
- NEW Hooks: `mpm_query`, `timed_callback`, and `get_name`
- CHANGED interfaces: `monitor hook`, `ap_reclaim_child_processes`, `ap_relieve_child_processes`

### **scoreboard (changed)**

`ap_get_scoreboard_worker` is made non-backwards-compatible as an alternative version is introduced. Additional `proxy_balancer` support. Child status stuff revamped.

### **util\_cookies (NEW!)**

Introduces a new API for managing HTTP Cookies.

### **util\_ldap (changed)**

*no description available*

### **util\_mutex (NEW!)**

A wrapper for APR proc and global mutexes in httpd, providing common configuration for the underlying mechanism and location of lock files.

## **util\_script (changed)**

NEW: ap\_args\_to\_table

## **util\_time (changed)**

NEW: ap\_recent\_ctime\_ex



## Logging

In order to take advantage of per-module loglevel configuration, any source file that calls the `ap_log_*` functions should declare which module it belongs to. If the module's `module_struct` is called `foo_module`, the following code can be used to remain backward compatible with HTTPD 2.0 and 2.2:

```
#include <http_log.h>

#ifdef APLOG_USE_MODULE
APLOG_USE_MODULE(foo);
#endif
```

Note: This is absolutely required for C++-language modules. It can be skipped for C-language modules, though that breaks module-specific log level support for files without it.

The number of parameters of the `ap_log_*` functions and the definition of `APLOG_MARK` has changed. Normally, the change is completely transparent. However, changes are required if a module uses `APLOG_MARK` as a parameter to its own functions or if a module calls `ap_log_*` without passing `APLOG_MARK`. A module which uses wrappers around `ap_log_*` typically uses both of these constructs.

The easiest way to change code which passes `APLOG_MARK` to its own functions is to define and use a different macro that expands to the parameters required by those functions, as `APLOG_MARK` should only be used when calling `ap_log_*` directly. In this way, the code will remain compatible with HTTPD 2.0 and 2.2.

Code which calls `ap_log_*` without passing `APLOG_MARK` will necessarily differ between 2.4 and earlier releases, as 2.4 requires

a new third argument, `APLOG_MODULE_INDEX`.

```
/* code for httpd 2.0/2.2 */
ap_log_perror(file, line, APLOG_ERR, 0, p, "Failed to allocate
dynamic lock structure");

/* code for httpd 2.4 */
ap_log_perror(file, line, APLOG_MODULE_INDEX, APLOG_ERR, 0, p,
"Failed to allocate dynamic lock structure");
```

`ap_log_*error` are now implemented as macros. This means that it is no longer possible to use `#ifdef` inside the argument list of `ap_log_*error`, as this would cause undefined behavior according to C99.

A `server_rec` pointer must be passed to `ap_log_error()` when called after startup. This was always appropriate, but there are even more limitations with a `NULL` `server_rec` in 2.4 than in previous releases. Beginning with 2.3.12, the global variable `ap_server_conf` can always be used as the `server_rec` parameter, as it will be `NULL` only when it is valid to pass `NULL` to `ap_log_error()`. `ap_server_conf` should be used only when a more appropriate `server_rec` is not available.

Consider the following changes to take advantage of the new `APLOG_TRACE1` . . . 8 log levels:

- Check current use of `APLOG_DEBUG` and consider if one of the `APLOG_TRACEn` levels is more appropriate.
- If your module currently has a mechanism for configuring the amount of debug logging which is performed, consider eliminating that mechanism and relying on the use of different `APLOG_TRACEn` levels. If expensive trace processing needs to be bypassed depending on the configured log level, use the `APLOGtracen` and `APLOGrtracen` macros to first check if

tracing is enabled.

Modules sometimes add process id and/or thread id to their log messages. These ids are now logged by default, so it may not be necessary for the module to log them explicitly. (Users may remove them from the error log format, but they can be instructed to add it back if necessary for problem diagnosis.)

## **If your module uses these existing APIs...**

### **ap\_default\_type()**

This is no longer available; Content-Type must be configured explicitly or added by the application.

### **ap\_get\_server\_name()**

If the returned server name is used in a URL, use `ap_get_server_name_for_url()` instead. This new function handles the odd case where the server name is an IPv6 literal address.

### **ap\_get\_server\_version()**

For logging purposes, where detailed information is appropriate, use `ap_get_server_description()`. When generating output, where the amount of information should be configurable by ServerTokens, use `ap_get_server_banner()`.

### **ap\_graceful\_stop\_signalled()**

Replace with a call to `ap_mpm_query(AP_MPMQ_MPM_STATE)` and checking for state `AP_MPMQ_STOPPING`.

### **ap\_max\_daemons\_limit, ap\_my\_generation, and ap\_threads\_per\_child**

Use `ap_mpm_query()` query codes `AP_MPMQ_MAX_DAEMON_USED`, `AP_MPMQ_GENERATION`, and `AP_MPMQ_MAX_THREADS`, respectively.

## **ap\_mpm\_query()**

Ensure that it is not used until after the register-hooks hook has completed. Otherwise, an MPM built as a DSO would not have had a chance to enable support for this function.

## **ap\_requires()**

The core server now provides better infrastructure for handling [Require](#) configuration. Register an auth provider function for each supported entity using `ap_register_auth_provider()`. The function will be called as necessary during [Require](#) processing. (Consult bundled modules for detailed examples.)

## **ap\_server\_conf->process->pool\_userdata**

Optional:

- If your module uses this to determine which pass of the startup hooks is being run, use `ap_state_query(AP_SQ_MAIN_STATE)`.
- If your module uses this to maintain data across the unloading and reloading of your module, use `ap_retained_data_create()` and `ap_retained_data_get()`.

## **apr\_global\_mutex\_create(), apr\_proc\_mutex\_create()**

Optional: See `ap_mutex_register()`, `ap_global_mutex_create()`, and `ap_proc_mutex_create()`; these allow your mutexes to be configurable with the [Mutex](#) directive; you can also remove any configuration mechanisms in your module for such mutexes

## **CORE\_PRIVATE**

This is now unnecessary and ignored.

## **dav\_new\_error() and dav\_new\_error\_tag()**

Previously, these assumed that `errno` contained information

describing the failure. Now, an `apr_status_t` parameter must be provided. Pass `0/APR_SUCCESS` if there is no such error information, or a valid `apr_status_t` value otherwise.

**`mpm_default.h`, `DEFAULT_LOCKFILE`,  
`DEFAULT_THREAD_LIMIT`, `DEFAULT_PIDLOG`, etc.**

The header file and most of the default configuration values set in it are no longer visible to modules. (Most can still be overridden at build time.) `DEFAULT_PIDLOG` and `DEFAULT_REL_RUNTIMEDIR` are now universally available via `ap_config.h`.

**`unixd_config`**

This has been renamed to `ap_unixd_config`.

**`unixd_setup_child()`**

This has been renamed to `ap_unixd_setup_child()`, but most callers should call the added `ap_run_drop_privileges()` hook.

**`conn_rec->remote_ip` and `conn_rec->remote_addr`**

These fields have been renamed in order to distinguish between the client IP address of the connection and the useragent IP address of the request (potentially overridden by a load balancer or proxy). References to either of these fields must be updated with one of the following options, as appropriate for the module:

- When you require the IP address of the user agent, which might be connected directly to the server, or might optionally be separated from the server by a transparent load balancer or proxy, use `request_rec->useragent_ip` and `request_rec->useragent_addr`.
- When you require the IP address of the client that is connected directly to the server, which might be the useragent or might be the load balancer or proxy itself, use `conn_rec->client_ip` and `conn_rec->`

>client\_addr.

## If your module interfaces with this feature...

### suEXEC

Optional: If your module logs an error when `ap_unixd_config.suexec_enabled` is 0, also log the value of the new field `suexec_disabled_reason`, which contains an explanation of why it is not available.

### Extended status data in the scoreboard

In previous releases, `ExtendedStatus` had to be set to `On`, which in turn required that `mod_status` was loaded. In 2.4, just set `ap_extended_status` to `1` in a pre-config hook and the extended status data will be available.

## Does your module...

### Parse query args

Consider if `ap_args_to_table()` would be helpful.

### Parse form data...

Use `ap_parse_form_data()`.

### Check for request header fields **Content - Length** and **Transfer - Encoding** to see if a body was specified

Use `ap_request_has_body()`.

### Implement cleanups which clear pointer variables

Use `ap_pool_cleanup_set_null()`.

### Create run-time files such as shared memory files, pid files, etc.

Use `ap_runtime_dir_relative()` so that the global configuration for the location of such files, either by the `DEFAULT_REL_RUNTIMEDIR` compile setting or the [DefaultRuntimeDir](#) directive, will be respected. *Apache*

*httpd 2.4.2 and above.*

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# Expressions in Apache HTTP Server

Historically, there are several syntax variants for expressions used to express a condition in the different modules of the Apache HTTP Server. There is some ongoing effort to only use a single variant, called *ap\_expr*, for all configuration directives. This document describes the *ap\_expr* expression parser.

The *ap\_expr* expression is intended to replace most other expression variants in HTTPD. For example, the deprecated [SSLRequire](#) expressions can be replaced by [Require expr](#).



## See also

[<If>](#)

[<ElseIf>](#)

[<Else>](#)

[ErrorDocument](#)

[Alias](#)

[ScriptAlias](#)

[Redirect](#)

[AuthBasicFake](#)

[AuthFormLoginRequiredLocation](#)

[AuthFormLoginSuccessLocation](#)

[AuthFormLogoutLocation](#)

[RewriteCond](#)

[SetEnvIfExpr](#)

[Header](#)

[RequestHeader](#)  
[FilterProvider](#)  
[Require expr](#)  
[Require ldap-user](#)  
[Require ldap-group](#)  
[Require ldap-dn](#)  
[Require ldap-attribute](#)  
[Require ldap-filter](#)  
[Require dbd-group](#)  
[Require dbm-group](#)  
[Require group](#)  
[Require host](#)  
[SSLRequire](#)  
[LogMessage](#)  
[mod\\_include](#)



[Backus-Naur Form](#) (BNF) is a notation technique for context-free grammars, often used to describe the syntax of languages used in computing. In most cases, expressions are used to express boolean values. For these, the starting point in the BNF is `expr`. However, a few directives like [LogMessage](#) accept expressions that evaluate to a string value. For those, the starting point in the BNF is `string`.

```
expr ::= "true" | "false"
      | "!" expr
      | expr "&&" expr
      | expr "||" expr
      | "(" expr ")"
      | comp
```

```
comp ::= stringcomp
      | integercomp
      | unaryop word
      | word binaryop word
      | word "in" "{" wordlist "}"
      | word "in" listfunction
      | word "=~" regex
      | word "!~" regex
```

```
stringcomp ::= word "==" word
            | word "!=" word
            | word "<" word
            | word "<=" word
            | word ">" word
            | word ">=" word
```

```
integercomp ::= word "-eq" word | word "eq" word
            | word "-ne" word | word "ne" word
            | word "-lt" word | word "lt" word
```

```

| word "-le" word | word "le" word
| word "-gt" word | word "gt" word
| word "-ge" word | word "ge" word

wordlist ::= word
          | wordlist "," word

word ::= word "." word
      | digit
      | "'" string "'"
      | "\"" string "\""
      | variable
      | rebackref
      | function

string ::= stringpart
        | string stringpart

stringpart ::= cstring
            | variable
            | rebackref

cstring ::= ...
digit   ::= [0-9]+

variable ::= "%{" varname "}"
          | "%{" funcname ":" funcargs "}"

rebackref ::= "$" [0-9]

function ::= funcname "(" word ")"

listfunction ::= listfuncname "(" word ")"

```



The expression parser provides a number of variables of the form `%{HTTP_HOST}`. Note that the value of a variable may depend on the phase of the request processing in which it is evaluated. For example, an expression used in an `<If >` directive is evaluated before authentication is done. Therefore, `%{REMOTE_USER}` will not be set in this case.

The following variables provide the values of the named HTTP request headers. The values of other headers can be obtained with the `req` [function](#). Using these variables may cause the header name to be added to the Vary header of the HTTP response, except where otherwise noted for the directive accepting the expression. The `req_novary` [function](#) may be used to circumvent this behavior.

Name
HTTP_ACCEPT
HTTP_COOKIE
HTTP_FORWARDED
HTTP_HOST
HTTP_PROXY_CONNECTION
HTTP_REFERER
HTTP_USER_AGENT

Other request related variables

Name	Description
REQUEST_METHOD	The HTTP method of the incoming request (e.g. GET)
REQUEST_SCHEME	The scheme part of the request's URI

REQUEST_URI	The path part of the request's URI
DOCUMENT_URI	Same as REQUEST_URI
REQUEST_FILENAME	The full local filesystem path to the file or script matching the request, if this has already been determined by the server at the time REQUEST_FILENAME is referenced. Otherwise, such as when used in virtual host context, the same value as REQUEST_URI
SCRIPT_FILENAME	Same as REQUEST_FILENAME
LAST_MODIFIED	The date and time of last modification of the file in the format 20101231235959, if this has already been determined by the server at the time LAST_MODIFIED is referenced.
SCRIPT_USER	The user name of the owner of the script.
SCRIPT_GROUP	The group name of the group of the script.
PATH_INFO	The trailing path name information, see <a href="#">AcceptPathInfo</a>
QUERY_STRING	The query string of the current request
IS_SUBREQ	"true" if the current request is a subrequest, "false" otherwise
THE_REQUEST	The complete request line (e.g., "GET /index.html HTTP/1.1")
REMOTE_ADDR	The IP address of the remote host
REMOTE_PORT	The port of the remote host (2.4.26)

	and later)
REMOTE_HOST	The host name of the remote host
REMOTE_USER	The name of the authenticated user, if any (not available during <code>&lt;If &gt;</code> )
REMOTE_IDENT	The user name set by <code>mod_ident</code>
SERVER_NAME	The <code>ServerName</code> of the current vhost
SERVER_PORT	The server port of the current vhost, see <code>ServerName</code>
SERVER_ADMIN	The <code>ServerAdmin</code> of the current vhost
SERVER_PROTOCOL	The protocol used by the request
DOCUMENT_ROOT	The <code>DocumentRoot</code> of the current vhost
AUTH_TYPE	The configured <code>AuthType</code> (e.g. "basic")
CONTENT_TYPE	The content type of the response (not available during <code>&lt;If &gt;</code> )
HANDLER	The name of the <code>handler</code> creating the response
HTTP2	"on" if the request uses http/2, "off" otherwise
HTTPS	"on" if the request uses https, "off" otherwise
IPV6	"on" if the connection uses IPv6, "off" otherwise
REQUEST_STATUS	The HTTP error status of the request (not available during <code>&lt;If &gt;</code> )

REQUEST_LOG_ID	The error log id of the request (see <a href="#">ErrorLogFormat</a> )
CONN_LOG_ID	The error log id of the connection (see <a href="#">ErrorLogFormat</a> )
CONN_REMOTE_ADDR	The peer IP address of the connection (see the <a href="#">mod_remoteip</a> module)
CONTEXT_PREFIX	
CONTEXT_DOCUMENT_ROOT	

### Misc variables

Name	Description
TIME_YEAR	The current year (e.g. 2010)
TIME_MON	The current month (01, ..., 12)
TIME_DAY	The current day of the month (01, ...)
TIME_HOUR	The hour part of the current time (00, ..., 23)
TIME_MIN	The minute part of the current time
TIME_SEC	The second part of the current time
TIME_WDAY	The day of the week (starting with 0 for Sunday)
TIME	The date and time in the format 20101231235959
SERVER_SOFTWARE	The server version string
API_VERSION	The date of the API version (module magic number)

Some modules register additional variables, see e.g. [mod\\_ssl](#).



## Binary operators

With the exception of some built-in comparison operators, binary operators have the form "- [a-zA-Z][a-zA-Z0-9\_]+", i.e. a minus and at least two characters. The name is not case sensitive. Modules may register additional binary operators.

### Comparison operators

Name	Alternative	Description
==	=	String equality
!=		String inequality
<		String less than
<=		String less than or equal
>		String greater than
>=		String greater than or equal
=~		String matches the regular expression
!~		String does not match the regular expression
-eq	eq	Integer equality
-ne	ne	Integer inequality
-lt	lt	Integer less than
-le	le	Integer less than or equal
-gt	gt	Integer greater than
-ge	ge	Integer greater than or equal

### Other binary operators

Name	Description
-ipmatch	IP address matches address/netmask
-strmatch	left string matches pattern given by right string (containing wildcards *, ?, [])
-	same as -strmatch, but case insensitive

strcmatch	
-fnmatch	same as -strmatch, but slashes are not matched by wildcards



## Unary operators

Unary operators take one argument and have the form "- [a-zA-Z]", i.e. a minus and one character. The name *is* case sensitive. Modules may register additional unary operators.

Name	Description	Restricted
-d	The argument is treated as a filename. True if the file exists and is a directory	yes
-e	The argument is treated as a filename. True if the file (or dir or special) exists	yes
-f	The argument is treated as a filename. True if the file exists and is regular file	yes
-s	The argument is treated as a filename. True if the file exists and is not empty	yes
-L	The argument is treated as a filename. True if the file exists and is symlink	yes
-h	The argument is treated as a filename. True if the file exists and is symlink (same as -L)	yes
-F	True if string is a valid file, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!	
-U	True if string is a valid URL, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!	
-A	Alias for -U	
-n	True if string is not empty	

-z	True if string is empty	
-T	False if string is empty, "0", "off", "false", or "no" (case insensitive). True otherwise.	
-R	Same as "%{REMOTE_ADDR} -ipmatch . . .", but more efficient	

The operators marked as "restricted" are not available in some modules like [mod\\_include](#).



Normal string-valued functions take one string as argument and return a string. Functions names are not case sensitive. Modules may register additional functions.

Name	Description	Special notes
req, http	Get HTTP request header; header names may be added to the Vary header, see below	
req_novary	Same as req, but header names will not be added to the Vary header	
resp	Get HTTP response header	
reqenv	Lookup request environment variable (as a shortcut, v can also be used to access variables).	ordering
osenv	Lookup operating system environment variable	
note	Lookup request note	ordering
env	Return first match of note, reqenv, osenv	ordering
tolower	Convert string to lower case	
toupper	Convert string to upper case	
escape	Escape special characters in %hex encoding	
unescape	Unescape %hex encoded string, leaving encoded slashes alone; return empty string if %00 is found	
base64	Encode the string using base64 encoding	
unbase64	Decode base64 encoded string, return truncated string if 0x00 is found	

md5	Hash the string using MD5, then encode the hash with hexadecimal encoding	
sha1	Hash the string using SHA1, then encode the hash with hexadecimal encoding	
file	Read contents from a file (including line endings, when present)	restricted
filemod	Return last modification time of a file (or 0 if file does not exist or is not regular file)	restricted
filesize	Return size of a file (or 0 if file does not exist or is not regular file)	restricted

The functions marked as "restricted" in the final column are not available in some modules like [mod\\_include](#).

The functions marked as "ordering" in the final column require some consideration for the ordering of different components of the server, especially when the function is used within the `<If>` directive which is evaluated relatively early.

### Environment variable ordering

When environment variables are looked up within an `<If>` condition, it's important to consider how extremely early in request processing that this resolution occurs. As a guideline, any directive defined outside of virtual host context (directory, location, htaccess) is not likely to have yet had a chance to execute. `SetEnvIf` in virtual host scope is one directive that runs prior to this resolution

When `reqenv` is used outside of `<If>`, the resolution will generally occur later, but the exact timing depends on the directive the expression has been used within.

When the functions `req` or `http` are used, the header name will automatically be added to the Vary header of the HTTP response, except where otherwise noted for the directive accepting the expression. The `req_novary` function can be used to prevent names from being added to the Vary header.

In addition to string-valued functions, there are also list-valued functions which take one string as argument and return a wordlist, i.e. a list of strings. The wordlist can be used with the special `-in` operator. Functions names are not case sensitive. Modules may register additional functions.

There are no built-in list-valued functions. `mod_ssl` provides `PeerExtList`. See the description of [SSLRequire](#) for details (but `PeerExtList` is also usable outside of [SSLRequire](#)).



The following examples show how expressions might be used to evaluate requests:

```
# Compare the host name to example.com and
<If "%{HTTP_HOST} == 'example.com'">
    Redirect permanent "/" "http://www.example.com"
</If>

# Force text/plain if requesting a file with
<If "%{QUERY_STRING} =~ /forcetext/">
    ForceType text/plain
</If>

# Only allow access to this content during business hours
<Directory "/foo/bar/business">
    Require expr %{TIME_HOUR} -gt 9 && %{TIME_HOUR} -lt 18
</Directory>

# Check a HTTP header for a list of values
<If "%{HTTP:X-example-header} in { 'foo', 'bar', 'baz' }">
    Header set matched true
</If>

# Check an environment variable for a regular expression
<If "! reqenv('REDIRECT_FOO') =~ /bar/">
    Header set matched true
</If>

# Check result of URI mapping by running in a directory
<Directory "/var/www">
    AddEncoding x-gzip gz
    <If "-f '%{REQUEST_FILENAME}.unzipme' && ! %{">
        SetOutputFilter INFLATE
    </If>
</Directory>
```

```
# Check against the client IP
<If "-R '192.168.1.0/24'">
    Header set matched true
</If>

# Function example in boolean context
<If "md5('foo') == 'acbd18db4cc2f85cedef6541'">
    Header set checksum-matched true
</If>

# Function example in string context
Header set foo-checksum "expr=%{md5:foo}"

# This delays the evaluation of the condition
Header always set CustomHeader my-value "expr=%{md5:foo}"
```



Name	Alternative	Description
-in	in	string contained in wordlist
/regexp/	m#regexp#	Regular expression (the second form allows different delimiters than /)
/regexp/i	m#regexp#i	Case insensitive regular expression
\$0 ... \$9		Regular expression backreferences

## Regular expression backreferences

The strings \$0 ... \$9 allow to reference the capture groups from a previously executed, successfully matching regular expressions. They can normally only be used in the same expression as the matching regex, but some modules allow special uses.



---

The *ap\_expr* syntax is mostly a superset of the syntax of the deprecated [SSLRequire](#) directive. The differences are described in [SSLRequire](#)'s documentation.



The req\_novary [function](#) is available for versions 2.4.4 and later.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

--



Microsoft Windows 2.0 , , .

: [Microsoft Windows](#)

.

: [Microsoft Windows](#)



---

## Novell NetWare

Novell NetWare 5.1 2.0

: [Novell NetWare](#)

## EBCDIC

1.3 EBCDIC

(-ASCII)

: 2.0 . ,  
.

: [EBCDIC](#)



| | [FAQ](#) | |

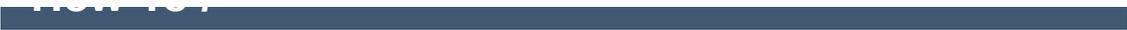


HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## How-To /





(authentication) . (authorization)

.  
:  
: [.htaccess](#)

**CGI**

CGI (Common Gateway Interface) CGI CGI  
, ( ) .  
. CGI , CGI .  
:  
: [CGI:](#)

**.htaccess**

.htaccess .  
, .  
:  
: [.htaccess](#)

**Server Side Includes**

SSI (Server Side Includes) HTML , .  
SSI CGI HTML  
.  
:  
: [Server Side Includes \(SSI\)](#)

UserDir

URL <http://example.com/~username/>  
"username" UserDir  
:  
: [\(public\\_html\)](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## suexec -

CGI suexec .  
root suexec setuid  
root .

suexec suexec

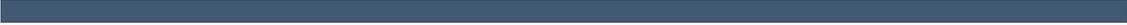
<http://httpd.apache.org/docs/2.4/suexec.html> .



---

**suexec -V**





**-v**

root suexec .

.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## Override Class Index for .htaccess

This is an index of the directives that are allowed in .htaccess files for various [AllowOverride](#) settings, organized by class. Its intended purpose is to help server administrators verify the privileges they're granting to .htaccess users. For an overview of how .htaccess works, see the [.htaccess tutorial](#).

To determine the set of directives that your server configuration allows .htaccess users to use:

1. Start with the set of directives in the `AllowOverrideList` for the directory in question. (By default, this is set to `None`.)
2. Find the `AllowOverride` setting for the directory in question. (By default, it is set to `None`.) There are two special cases:
  1. If your `AllowOverride` setting is `All`, add every directive listed on this page to the list.
  2. If your `AllowOverride` setting is `None`, you're done. Only the directives in the `AllowOverrideList` (if any) will be allowed.
3. For each override class listed in `AllowOverride`, look up the corresponding set of directives below and add them to the list.
4. Finally, add the set of directives that is always allowed in .htaccess (these are listed in the [All section](#), below).

Several of the override classes are quite powerful and give .htaccess users a large amount of control over the server. For a stricter approach, set `AllowOverride None` and use [AllowOverrideList](#) to specify the exact list of directives that .htaccess users are allowed to use.



The following directives are allowed in any .htaccess file, as long as overrides are enabled in the server configuration.

<a href="#">&lt;Else&gt;</a>	<a href="#">core</a>
Contains directives that apply only if the condition of a previous <a href="#">&lt;If&gt;</a> or <a href="#">&lt;ElseIf&gt;</a> section is not satisfied by a request at runtime	
<a href="#">&lt;Elseif&gt;</a>	<a href="#">core</a>
Contains directives that apply only if a condition is satisfied by a request at runtime while the condition of a previous <a href="#">&lt;If&gt;</a> or <a href="#">&lt;ElseIf&gt;</a> section is not satisfied	
<a href="#">&lt;Files&gt;</a>	<a href="#">core</a>
Contains directives that apply to matched filenames	
<a href="#">&lt;FilesMatch&gt;</a>	<a href="#">core</a>
Contains directives that apply to regular-expression matched filenames	
<a href="#">&lt;If&gt;</a>	<a href="#">core</a>
Contains directives that apply only if a condition is satisfied by a request at runtime	
<a href="#">&lt;IfDefine&gt;</a>	<a href="#">core</a>
Encloses directives that will be processed only if a test is true at startup	
<a href="#">&lt;IfModule&gt;</a>	<a href="#">core</a>
Encloses directives that are processed conditional on the presence or absence of a specific module	
<a href="#">&lt;IfVersion&gt;</a>	<a href="#">mod_version</a>
contains version dependent configuration	
<a href="#">LimitRequestBody</a>	<a href="#">core</a>
Restricts the total size of the HTTP request body sent from the client	
<a href="#">LimitXMLRequestBody</a>	<a href="#">core</a>
Limits the size of an XML-based request body	
<a href="#">LuaCodeCache</a>	<a href="#">mod_lua</a>
Configure the compiled code cache.	
<a href="#">LuaHookAccessChecker</a>	<a href="#">mod_lua</a>
Provide a hook for the access_checker phase of request processing	
<a href="#">LuaHookAuthChecker</a>	<a href="#">mod_lua</a>
Provide a hook for the auth_checker phase of request processing	
<a href="#">LuaHookCheckUserID</a>	<a href="#">mod_lua</a>
Provide a hook for the check_user_id phase of request processing	
<a href="#">LuaHookFixups</a>	<a href="#">mod_lua</a>

Provide a hook for the fixups phase of a request processing	
<a href="#">LuaHookInsertFilter</a>	<a href="#">mod_lua</a>
Provide a hook for the insert_filter phase of request processing	
<a href="#">LuaHookLog</a>	<a href="#">mod_lua</a>
Provide a hook for the access log phase of a request processing	
<a href="#">LuaHookMapToStorage</a>	<a href="#">mod_lua</a>
Provide a hook for the map_to_storage phase of request processing	
<a href="#">LuaHookTranslateName</a>	<a href="#">mod_lua</a>
Provide a hook for the translate name phase of request processing	
<a href="#">LuaHookTypeChecker</a>	<a href="#">mod_lua</a>
Provide a hook for the type_checker phase of request processing	
<a href="#">LuaInherit</a>	<a href="#">mod_lua</a>
Controls how parent configuration sections are merged into children	
<a href="#">LuaMapHandler</a>	<a href="#">mod_lua</a>
Map a path to a lua handler	
<a href="#">LuaPackageCPath</a>	<a href="#">mod_lua</a>
Add a directory to lua's package.cpath	
<a href="#">LuaPackagePath</a>	<a href="#">mod_lua</a>
Add a directory to lua's package.path	
<a href="#">LuaQuickHandler</a>	<a href="#">mod_lua</a>
Provide a hook for the quick handler of request processing	
<a href="#">LuaRoot</a>	<a href="#">mod_lua</a>
Specify the base path for resolving relative paths for mod_lua directives	
<a href="#">LuaScope</a>	<a href="#">mod_lua</a>
One of once, request, conn, thread -- default is once	
<a href="#">RLimitCPU</a>	<a href="#">core</a>
Limits the CPU consumption of processes launched by Apache httpd children	
<a href="#">RLimitMEM</a>	<a href="#">core</a>
Limits the memory consumption of processes launched by Apache httpd children	
<a href="#">RLimitNPROC</a>	<a href="#">core</a>
Limits the number of processes that can be launched by processes launched by Apache httpd children	
<a href="#">ServerSignature</a>	<a href="#">core</a>
Configures the footer on server-generated documents	
<a href="#">SSIErrorMsg</a>	<a href="#">mod_include</a>

Error message displayed when there is an SSI error	
<a href="#">SSITimeFormat</a>	<a href="#">mod_include</a>
Configures the format in which date strings are displayed	
<a href="#">SSIUndefinedEcho</a>	<a href="#">mod_include</a>
String displayed when an unset variable is echoed	



The following directives are allowed in .htaccess files when AllowOverride AuthConfig is in effect. They give .htaccess users control over the authentication and authorization methods that are applied to their directory subtrees, including several related utility directives for session handling and TLS settings.

<a href="#">Anonymous</a>	<a href="#">mod_auth_anon</a>
Specifies userIDs that are allowed access without password verification	
<a href="#">Anonymous_LogEmail</a>	<a href="#">mod_auth_anon</a>
Sets whether the password entered will be logged in the error log	
<a href="#">Anonymous_MustGiveEmail</a>	<a href="#">mod_auth_anon</a>
Specifies whether blank passwords are allowed	
<a href="#">Anonymous_NoUserID</a>	<a href="#">mod_auth_anon</a>
Sets whether the userID field may be empty	
<a href="#">Anonymous_VerifyEmail</a>	<a href="#">mod_auth_anon</a>
Sets whether to check the password field for a correctly formatted email address	
<a href="#">AuthBasicAuthoritative</a>	<a href="#">mod_auth_basic</a>
Sets whether authorization and authentication are passed to lower level modules	
<a href="#">AuthBasicFake</a>	<a href="#">mod_auth_basic</a>
Fake basic authentication using the given expressions for username and password	
<a href="#">AuthBasicProvider</a>	<a href="#">mod_auth_basic</a>
Sets the authentication provider(s) for this location	
<a href="#">AuthBasicUseDigestAlgorithm</a>	<a href="#">mod_auth_basic</a>
Check passwords against the authentication providers as if Digest Authentication was in force instead of Basic Authentication.	
<a href="#">AuthDBMGroupFile</a>	<a href="#">mod_authz_dbm</a>
Sets the name of the database file containing the list of user groups for authorization	
<a href="#">AuthDBMType</a>	<a href="#">mod_authn_dbm</a>
Sets the type of database file that is used to store passwords	
<a href="#">AuthDBMUserFile</a>	<a href="#">mod_authn_dbm</a>
Sets the name of a database file containing the list of users and passwords for authentication	
<a href="#">AuthDigestAlgorithm</a>	<a href="#">mod_auth_digest</a>
Selects the algorithm used to calculate the challenge and response hashes in digest authentication	

<a href="#">AuthDigestDomain</a>	<a href="#">mod_auth_digest</a>
URIs that are in the same protection space for digest authentication	
<a href="#">AuthDigestNonceLifetime</a>	<a href="#">mod_auth_digest</a>
How long the server nonce is valid	
<a href="#">AuthDigestProvider</a>	<a href="#">mod_auth_digest</a>
Sets the authentication provider(s) for this location	
<a href="#">AuthDigestQop</a>	<a href="#">mod_auth_digest</a>
Determines the quality-of-protection to use in digest authentication	
<a href="#">AuthFormAuthoritative</a>	<a href="#">mod_auth_form</a>
Sets whether authorization and authentication are passed to lower level modules	
<a href="#">AuthFormProvider</a>	<a href="#">mod_auth_form</a>
Sets the authentication provider(s) for this location	
<a href="#">AuthGroupFile</a>	<a href="#">mod_authz_groupfile</a>
Sets the name of a text file containing the list of user groups for authorization	
<a href="#">AuthLDAPAuthorizePrefix</a>	<a href="#">mod_authnz_ldap</a>
Specifies the prefix for environment variables set during authorization	
<a href="#">AuthLDAPBindAuthoritative</a>	<a href="#">mod_authnz_ldap</a>
Determines if other authentication providers are used when a user can be mapped to a DN but the server cannot successfully bind with the user's credentials.	
<a href="#">AuthLDAPBindDN</a>	<a href="#">mod_authnz_ldap</a>
Optional DN to use in binding to the LDAP server	
<a href="#">AuthLDAPBindPassword</a>	<a href="#">mod_authnz_ldap</a>
Password used in conjunction with the bind DN	
<a href="#">AuthLDAPCompareAsUser</a>	<a href="#">mod_authnz_ldap</a>
Use the authenticated user's credentials to perform authorization comparisons	
<a href="#">AuthLDAPCompareDNOnServer</a>	<a href="#">mod_authnz_ldap</a>
Use the LDAP server to compare the DNs	
<a href="#">AuthLDAPDereferenceAliases</a>	<a href="#">mod_authnz_ldap</a>
When will the module de-reference aliases	
<a href="#">AuthLDAPGroupAttribute</a>	<a href="#">mod_authnz_ldap</a>
LDAP attributes used to identify the user members of groups.	
<a href="#">AuthLDAPGroupAttributeIsDN</a>	<a href="#">mod_authnz_ldap</a>
Use the DN of the client username when checking for group membership	
<a href="#">AuthLDAPInitialBindAsUser</a>	<a href="#">mod_authnz_ldap</a>
Determines if the server does the initial DN lookup using the basic authentication	

users' own username, instead of anonymously or with hard-coded credentials for the server

### [AuthLDAPInitialBindPattern](#)

[mod\\_authnz\\_ldap](#)

Specifies the transformation of the basic authentication username to be used when binding to the LDAP server to perform a DN lookup

### [AuthLDAPMaxSubGroupDepth](#)

[mod\\_authnz\\_ldap](#)

Specifies the maximum sub-group nesting depth that will be evaluated before the user search is discontinued.

### [AuthLDAPRemoteUserAttribute](#)

[mod\\_authnz\\_ldap](#)

Use the value of the attribute returned during the user query to set the REMOTE\_USER environment variable

### [AuthLDAPRemoteUserIsDN](#)

[mod\\_authnz\\_ldap](#)

Use the DN of the client username to set the REMOTE\_USER environment variable

### [AuthLDAPSearchAsUser](#)

[mod\\_authnz\\_ldap](#)

Use the authenticated user's credentials to perform authorization searches

### [AuthLDAPSubGroupAttribute](#)

[mod\\_authnz\\_ldap](#)

Specifies the attribute labels, one value per directive line, used to distinguish the members of the current group that are groups.

### [AuthLDAPSubGroupClass](#)

[mod\\_authnz\\_ldap](#)

Specifies which LDAP objectClass values identify directory objects that are groups during sub-group processing.

### [AuthLDAPUrl](#)

[mod\\_authnz\\_ldap](#)

URL specifying the LDAP search parameters

### [AuthMerging](#)

[mod\\_authz\\_core](#)

Controls the manner in which each configuration section's authorization logic is combined with that of preceding configuration sections.

### [AuthName](#)

[mod\\_authn\\_core](#)

Authorization realm for use in HTTP authentication

### [AuthnCacheProvideFor](#)

[mod\\_authn\\_socache](#)

Specify which authn provider(s) to cache for

### [AuthnCacheTimeout](#)

[mod\\_authn\\_socache](#)

Set a timeout for cache entries

### [AuthType](#)

[mod\\_authn\\_core](#)

Type of user authentication

### [AuthUserFile](#)

[mod\\_authn\\_file](#)

Sets the name of a text file containing the list of users and passwords for authentication

<a href="#">AuthzDBMType</a>	<a href="#">mod_authz_dbm</a>
Sets the type of database file that is used to store list of user groups	
<a href="#">CGIPassAuth</a>	<a href="#">core</a>
Enables passing HTTP authorization headers to scripts as CGI variables	
<a href="#">LDAPReferralHopLimit</a>	<a href="#">mod_ldap</a>
The maximum number of referral hops to chase before terminating an LDAP query.	
<a href="#">LDAPReferrals</a>	<a href="#">mod_ldap</a>
Enable referral chasing during queries to the LDAP server.	
<a href="#">&lt;Limit&gt;</a>	<a href="#">core</a>
Restrict enclosed access controls to only certain HTTP methods	
<a href="#">&lt;LimitExcept&gt;</a>	<a href="#">core</a>
Restrict access controls to all HTTP methods except the named ones	
<a href="#">Require</a>	<a href="#">mod_authz_core</a>
Tests whether an authenticated user is authorized by an authorization provider.	
<a href="#">&lt;RequireAll&gt;</a>	<a href="#">mod_authz_core</a>
Enclose a group of authorization directives of which none must fail and at least one must succeed for the enclosing directive to succeed.	
<a href="#">&lt;RequireAny&gt;</a>	<a href="#">mod_authz_core</a>
Enclose a group of authorization directives of which one must succeed for the enclosing directive to succeed.	
<a href="#">&lt;RequireNone&gt;</a>	<a href="#">mod_authz_core</a>
Enclose a group of authorization directives of which none must succeed for the enclosing directive to not fail.	
<a href="#">Satisfy</a>	<a href="#">mod_access_compat</a>
Interaction between host-level access control and user authentication	
<a href="#">Session</a>	<a href="#">mod_session</a>
Enables a session for the current directory or location	
<a href="#">SessionEnv</a>	<a href="#">mod_session</a>
Control whether the contents of the session are written to the <code>HTTP_SESSION</code> environment variable	
<a href="#">SessionHeader</a>	<a href="#">mod_session</a>
Import session updates from a given HTTP response header	
<a href="#">SessionInclude</a>	<a href="#">mod_session</a>
Define URL prefixes for which a session is valid	
<a href="#">SessionMaxAge</a>	<a href="#">mod_session</a>
Define a maximum age in seconds for a session	

<a href="#">SSLCipherSuite</a>	<a href="#">mod_ssl</a>
Cipher Suite available for negotiation in SSL handshake	
<a href="#">SSLProxyCipherSuite</a>	<a href="#">mod_ssl</a>
Cipher Suite available for negotiation in SSL proxy handshake	
<a href="#">SSLRenegBufferSize</a>	<a href="#">mod_ssl</a>
Set the size for the SSL renegotiation buffer	
<a href="#">SSLRequire</a>	<a href="#">mod_ssl</a>
Allow access only when an arbitrarily complex boolean expression is true	
<a href="#">SSLRequireSSL</a>	<a href="#">mod_ssl</a>
Deny access when SSL is not used for the HTTP request	
<a href="#">SSLUserName</a>	<a href="#">mod_ssl</a>
Variable name to determine user name	
<a href="#">SSLVerifyClient</a>	<a href="#">mod_ssl</a>
Type of Client Certificate verification	
<a href="#">SSLVerifyDepth</a>	<a href="#">mod_ssl</a>
Maximum depth of CA Certificates in Client Certificate verification	



The following directives are allowed in .htaccess files when AllowOverride FileInfo is in effect. They give .htaccess users a wide range of control over the responses and metadata given by the server.

<a href="#">AcceptPathInfo</a>	<a href="#">core</a>
Resources accept trailing pathname information	
<a href="#">Action</a>	<a href="#">mod_actions</a>
Activates a CGI script for a particular handler or content-type	
<a href="#">AddCharset</a>	<a href="#">mod_mime</a>
Maps the given filename extensions to the specified content charset	
<a href="#">AddDefaultCharset</a>	<a href="#">core</a>
Default charset parameter to be added when a response content-type is text/plain or text/html	
<a href="#">AddEncoding</a>	<a href="#">mod_mime</a>
Maps the given filename extensions to the specified encoding type	
<a href="#">AddHandler</a>	<a href="#">mod_mime</a>
Maps the filename extensions to the specified handler	
<a href="#">AddInputFilter</a>	<a href="#">mod_mime</a>
Maps filename extensions to the filters that will process client requests	
<a href="#">AddLanguage</a>	<a href="#">mod_mime</a>
Maps the given filename extension to the specified content language	
<a href="#">AddOutputFilter</a>	<a href="#">mod_mime</a>
Maps filename extensions to the filters that will process responses from the server	
<a href="#">AddOutputFilterByType</a>	<a href="#">mod_filter</a>
assigns an output filter to a particular media-type	
<a href="#">AddType</a>	<a href="#">mod_mime</a>
Maps the given filename extensions onto the specified content type	
<a href="#">BrowserMatch</a>	<a href="#">mod_setenvif</a>
Sets environment variables conditional on HTTP User-Agent	
<a href="#">BrowserMatchNoCase</a>	<a href="#">mod_setenvif</a>
Sets environment variables conditional on User-Agent without respect to case	
<a href="#">CGIMapExtension</a>	<a href="#">core</a>
Technique for locating the interpreter for CGI scripts	

<a href="#">CGIVar</a>	<a href="#">core</a>
Controls how some CGI variables are set	
<a href="#">CharsetDefault</a>	<a href="#">mod_charset_lite</a>
Charset to translate into	
<a href="#">CharsetOptions</a>	<a href="#">mod_charset_lite</a>
Configures charset translation behavior	
<a href="#">CharsetSourceEnc</a>	<a href="#">mod_charset_lite</a>
Source charset of files	
<a href="#">CookieDomain</a>	<a href="#">mod_usertrack</a>
The domain to which the tracking cookie applies	
<a href="#">CookieExpires</a>	<a href="#">mod_usertrack</a>
Expiry time for the tracking cookie	
<a href="#">CookieName</a>	<a href="#">mod_usertrack</a>
Name of the tracking cookie	
<a href="#">CookieStyle</a>	<a href="#">mod_usertrack</a>
Format of the cookie header field	
<a href="#">CookieTracking</a>	<a href="#">mod_usertrack</a>
Enables tracking cookie	
<a href="#">DefaultLanguage</a>	<a href="#">mod_mime</a>
Defines a default language-tag to be sent in the Content-Language header field for all resources in the current context that have not been assigned a language-tag by some other means.	
<a href="#">DefaultType</a>	<a href="#">core</a>
This directive has no effect other than to emit warnings if the value is not none. In prior versions, DefaultType would specify a default media type to assign to response content for which no other media type configuration could be found.	
<a href="#">EnableMMAP</a>	<a href="#">core</a>
Use memory-mapping to read files during delivery	
<a href="#">EnableSendfile</a>	<a href="#">core</a>
Use the kernel sendfile support to deliver files to the client	
<a href="#">ErrorDocument</a>	<a href="#">core</a>
What the server will return to the client in case of an error	
<a href="#">FileETag</a>	<a href="#">core</a>
File attributes used to create the ETag HTTP response header for static files	
<a href="#">ForceLanguagePriority</a>	<a href="#">mod_negotiation</a>
Action to take if a single acceptable document is not found	

<a href="#">ForceType</a>	<a href="#">core</a>
Forces all matching files to be served with the specified media type in the HTTP Content-Type header field	
<a href="#">Header</a>	<a href="#">mod_headers</a>
Configure HTTP response headers	
<a href="#">ISAPIAppendLogToErrors</a>	<a href="#">mod_isapi</a>
Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log	
<a href="#">ISAPIAppendLogToQuery</a>	<a href="#">mod_isapi</a>
Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field	
<a href="#">ISAPIFakeAsync</a>	<a href="#">mod_isapi</a>
Fake asynchronous support for ISAPI callbacks	
<a href="#">ISAPILogNotSupported</a>	<a href="#">mod_isapi</a>
Log unsupported feature requests from ISAPI extensions	
<a href="#">ISAPIReadAheadBuffer</a>	<a href="#">mod_isapi</a>
Size of the Read Ahead Buffer sent to ISAPI extensions	
<a href="#">LanguagePriority</a>	<a href="#">mod_negotiation</a>
The precedence of language variants for cases where the client does not express a preference	
<a href="#">MultiviewsMatch</a>	<a href="#">mod_mime</a>
The types of files that will be included when searching for a matching file with MultiViews	
<a href="#">PassEnv</a>	<a href="#">mod_env</a>
Passes environment variables from the shell	
<a href="#">QualifyRedirectURL</a>	<a href="#">core</a>
Controls whether the REDIRECT_URL environment variable is fully qualified	
<a href="#">Redirect</a>	<a href="#">mod_alias</a>
Sends an external redirect asking the client to fetch a different URL	
<a href="#">RedirectMatch</a>	<a href="#">mod_alias</a>
Sends an external redirect based on a regular expression match of the current URL	
<a href="#">RedirectPermanent</a>	<a href="#">mod_alias</a>
Sends an external permanent redirect asking the client to fetch a different URL	
<a href="#">RedirectTemp</a>	<a href="#">mod_alias</a>
Sends an external temporary redirect asking the client to fetch a different URL	

<a href="#">RemoveCharset</a>	<a href="#">mod_mime</a>
Removes any character set associations for a set of file extensions	
<a href="#">RemoveEncoding</a>	<a href="#">mod_mime</a>
Removes any content encoding associations for a set of file extensions	
<a href="#">RemoveHandler</a>	<a href="#">mod_mime</a>
Removes any handler associations for a set of file extensions	
<a href="#">RemoveInputFilter</a>	<a href="#">mod_mime</a>
Removes any input filter associations for a set of file extensions	
<a href="#">RemoveLanguage</a>	<a href="#">mod_mime</a>
Removes any language associations for a set of file extensions	
<a href="#">RemoveOutputFilter</a>	<a href="#">mod_mime</a>
Removes any output filter associations for a set of file extensions	
<a href="#">RemoveType</a>	<a href="#">mod_mime</a>
Removes any content type associations for a set of file extensions	
<a href="#">RequestHeader</a>	<a href="#">mod_headers</a>
Configure HTTP request headers	
<a href="#">RewriteBase</a>	<a href="#">mod_rewrite</a>
Sets the base URL for per-directory rewrites	
<a href="#">RewriteCond</a>	<a href="#">mod_rewrite</a>
Defines a condition under which rewriting will take place	
<a href="#">RewriteEngine</a>	<a href="#">mod_rewrite</a>
Enables or disables runtime rewriting engine	
<a href="#">RewriteOptions</a>	<a href="#">mod_rewrite</a>
Sets some special options for the rewrite engine	
<a href="#">RewriteRule</a>	<a href="#">mod_rewrite</a>
Defines rules for the rewriting engine	
<a href="#">ScriptInterpreterSource</a>	<a href="#">core</a>
Technique for locating the interpreter for CGI scripts	
<a href="#">SetEnv</a>	<a href="#">mod_env</a>
Sets environment variables	
<a href="#">SetEnvIf</a>	<a href="#">mod_setenvif</a>
Sets environment variables based on attributes of the request	
<a href="#">SetEnvIfExpr</a>	<a href="#">mod_setenvif</a>
Sets environment variables based on an ap_expr expression	
<a href="#">SetEnvIfNoCase</a>	<a href="#">mod_setenvif</a>

	Sets environment variables based on attributes of the request without respect to case
<a href="#">SetHandler</a>	<a href="#">core</a>
	Forces all matching files to be processed by a handler
<a href="#">SetInputFilter</a>	<a href="#">core</a>
	Sets the filters that will process client requests and POST input
<a href="#">SetOutputFilter</a>	<a href="#">core</a>
	Sets the filters that will process responses from the server
<a href="#">Substitute</a>	<a href="#">mod_substitute</a>
	Pattern to filter the response content
<a href="#">SubstituteInheritBefore</a>	<a href="#">mod_substitute</a>
	Change the merge order of inherited patterns
<a href="#">SubstituteMaxLineLength</a>	<a href="#">mod_substitute</a>
	Set the maximum line size
<a href="#">UnsetEnv</a>	<a href="#">mod_env</a>
	Removes variables from the environment



The following directives are allowed in .htaccess files when `AllowOverride Indexes` is in effect. They allow .htaccess users to control aspects of the directory index pages provided by the server, including autoindex generation.

<a href="#">AddAlt</a>	<a href="#">mod_autoindex</a>
Alternate text to display for a file, instead of an icon selected by filename	
<a href="#">AddAltByEncoding</a>	<a href="#">mod_autoindex</a>
Alternate text to display for a file instead of an icon selected by MIME-encoding	
<a href="#">AddAltByType</a>	<a href="#">mod_autoindex</a>
Alternate text to display for a file, instead of an icon selected by MIME content-type	
<a href="#">AddDescription</a>	<a href="#">mod_autoindex</a>
Description to display for a file	
<a href="#">AddIcon</a>	<a href="#">mod_autoindex</a>
Icon to display for a file selected by name	
<a href="#">AddIconByEncoding</a>	<a href="#">mod_autoindex</a>
Icon to display next to files selected by MIME content-encoding	
<a href="#">AddIconByType</a>	<a href="#">mod_autoindex</a>
Icon to display next to files selected by MIME content-type	
<a href="#">DefaultIcon</a>	<a href="#">mod_autoindex</a>
Icon to display for files when no specific icon is configured	
<a href="#">DirectoryCheckHandler</a>	<a href="#">mod_dir</a>
Toggle how this module responds when another handler is configured	
<a href="#">DirectoryIndex</a>	<a href="#">mod_dir</a>
List of resources to look for when the client requests a directory	
<a href="#">DirectoryIndexRedirect</a>	<a href="#">mod_dir</a>
Configures an external redirect for directory indexes.	
<a href="#">DirectorySlash</a>	<a href="#">mod_dir</a>
Toggle trailing slash redirects on or off	
<a href="#">ExpiresActive</a>	<a href="#">mod_expires</a>
Enables generation of Expires headers	
<a href="#">ExpiresByType</a>	<a href="#">mod_expires</a>
Value of the Expires header configured by MIME type	
<a href="#">ExpiresDefault</a>	<a href="#">mod_expires</a>

	Default algorithm for calculating expiration time	
<a href="#">FallbackResource</a>	Define a default URL for requests that don't map to a file	<a href="#">mod_dir</a>
<a href="#">HeaderName</a>	Name of the file that will be inserted at the top of the index listing	<a href="#">mod_autoindex</a>
<a href="#">ImapBase</a>	Default base for imagemap files	<a href="#">mod_imagemap</a>
<a href="#">ImapDefault</a>	Default action when an imagemap is called with coordinates that are not explicitly mapped	<a href="#">mod_imagemap</a>
<a href="#">ImapMenu</a>	Action if no coordinates are given when calling an imagemap	<a href="#">mod_imagemap</a>
<a href="#">IndexHeadInsert</a>	Inserts text in the HEAD section of an index page.	<a href="#">mod_autoindex</a>
<a href="#">IndexIgnore</a>	Adds to the list of files to hide when listing a directory	<a href="#">mod_autoindex</a>
<a href="#">IndexIgnoreReset</a>	Empties the list of files to hide when listing a directory	<a href="#">mod_autoindex</a>
<a href="#">IndexOptions</a>	Various configuration settings for directory indexing	<a href="#">mod_autoindex</a>
<a href="#">IndexOrderDefault</a>	Sets the default ordering of the directory index	<a href="#">mod_autoindex</a>
<a href="#">IndexStyleSheet</a>	Adds a CSS stylesheet to the directory index	<a href="#">mod_autoindex</a>
<a href="#">MetaDir</a>	Name of the directory to find CERN-style meta information files	<a href="#">mod_cern_meta</a>
<a href="#">MetaFiles</a>	Activates CERN meta-file processing	<a href="#">mod_cern_meta</a>
<a href="#">MetaSuffix</a>	File name suffix for the file containing CERN-style meta information	<a href="#">mod_cern_meta</a>
<a href="#">ReadmeName</a>	Name of the file that will be inserted at the end of the index listing	<a href="#">mod_autoindex</a>



The following directives are allowed in .htaccess files when AllowOverride Limit is in effect. This extremely narrow override type mostly allows the use of the legacy authorization directives provided by [mod\\_access\\_compat](#).

<a href="#">Allow</a>	<a href="#">mod_access_compat</a>
Controls which hosts can access an area of the server	
<a href="#">Deny</a>	<a href="#">mod_access_compat</a>
Controls which hosts are denied access to the server	
<a href="#">&lt;Limit&gt;</a>	<a href="#">core</a>
Restrict enclosed access controls to only certain HTTP methods	
<a href="#">&lt;LimitExcept&gt;</a>	<a href="#">core</a>
Restrict access controls to all HTTP methods except the named ones	
<a href="#">Order</a>	<a href="#">mod_access_compat</a>
Controls the default access state and the order in which <a href="#">Allow</a> and <a href="#">Deny</a> are evaluated.	



*[This section has no description. It's possible that the documentation is incomplete, or that the directives here have an incorrect or misspelled Override type. Please consider reporting this in the [comments section](#).]*

[LogIOTrackTTFB](#)

[mod\\_logio](#)

Enable tracking of time to first byte (TTFB)



*[This section has no description. It's possible that the documentation is incomplete, or that the directives here have an incorrect or misspelled Override type. Please consider reporting this in the [comments section](#).]*

<a href="#">AuthnCacheEnable</a> Enable Authn caching configured anywhere	<a href="#">mod_auth_socache</a>
<a href="#">AuthnCacheSOCache</a> Select socache backend provider to use	<a href="#">mod_auth_socache</a>



## Not applicable

*[This section has no description. It's possible that the documentation is incomplete, or that the directives here have an incorrect or misspelled Override type. Please consider reporting this in the [comments section](#).]*

<a href="#">SSLProxyMachineCertificateChainFile</a>	<a href="#">mod_ssl</a>
File of concatenated PEM-encoded CA certificates to be used by the proxy for choosing a certificate	
<a href="#">SSLProxyMachineCertificateFile</a>	<a href="#">mod_ssl</a>
File of concatenated PEM-encoded client certificates and keys to be used by the proxy	
<a href="#">SSLProxyMachineCertificatePath</a>	<a href="#">mod_ssl</a>
Directory of PEM-encoded client certificates and keys to be used by the proxy	



## Options

The following directives are allowed in .htaccess files when `AllowOverride Options` is in effect. They give .htaccess users access to `Options` and similar directives, as well as directives that control the filter chain.

<a href="#">CheckCaseOnly</a>	<a href="#">mod_spelling</a>
Limits the action of the spelling module to case corrections	
<a href="#">CheckSpelling</a>	<a href="#">mod_spelling</a>
Enables the spelling module	
<a href="#">ContentDigest</a>	<a href="#">core</a>
Enables the generation of Content -MD5 HTTP Response headers	
<a href="#">FilterChain</a>	<a href="#">mod_filter</a>
Configure the filter chain	
<a href="#">FilterDeclare</a>	<a href="#">mod_filter</a>
Declare a smart filter	
<a href="#">FilterProtocol</a>	<a href="#">mod_filter</a>
Deal with correct HTTP protocol handling	
<a href="#">FilterProvider</a>	<a href="#">mod_filter</a>
Register a content filter	
<a href="#">Options</a>	<a href="#">core</a>
Configures what features are available in a particular directory	
<a href="#">ReflectorHeader</a>	<a href="#">mod_reflector</a>
Reflect an input header to the output headers	
<a href="#">SSLOptions</a>	<a href="#">mod_ssl</a>
Configure various SSL engine run-time options	
<a href="#">SSLProxyProtocol</a>	<a href="#">mod_ssl</a>
Configure usable SSL protocol flavors for proxy usage	
<a href="#">XBitHack</a>	<a href="#">mod_include</a>
Parse SSI directives in files with the execute bit set	

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Miscellaneous Documentation](#)

## Password Formats

Notes about the password encryption formats generated and understood by Apache.



There are five formats that Apache recognizes for basic-authentication passwords. Note that not all formats work on every platform:

### **bcrypt**

"\$2y\$" + the result of the `crypt_blowfish` algorithm. See the APR source file [crypt\\_blowfish.c](#) for the details of the algorithm.

### **MD5**

"\$apr1\$" + the result of an Apache-specific algorithm using an iterated (1,000 times) MD5 digest of various combinations of a random 32-bit salt and the password. See the APR source file [apr\\_md5.c](#) for the details of the algorithm.

### **SHA1**

"{SHA}" + Base64-encoded SHA-1 digest of the password. Insecure.

### **CRYPT**

Unix only. Uses the traditional Unix `crypt(3)` function with a randomly-generated 32-bit salt (only 12 bits used) and the first 8 characters of the password. Insecure.

### **PLAIN TEXT (i.e. *unencrypted*)**

Windows & Netware only. Insecure.

## **Generating values with `htpasswd`**

### **bcrypt**

```
$ htpasswd -nbB myName myPassword  
myName:$2y$05$c4WoMPo3SXsafkva.HHa6uXQZWr7oboPiC2bT/r7q1BB8I2s0BF
```

### **MD5**

```
$ htpasswd -nbm myName myPassword  
myName:$apr1$r31.....$HqJZimcKQFAMYayBlzkrA/
```

## SHA1

```
$ htpasswd -nbs myName myPassword  
myName:{SHA}VBPuJHI7uixaa6LQGwx4s+5GKNE=
```

## CRYPT

```
$ htpasswd -nbd myName myPassword  
myName:rqXexS6ZhobKA
```

## Generating CRYPT and MD5 values with the OpenSSL command-line program

OpenSSL knows the Apache-specific MD5 algorithm.

## MD5

```
$ openssl passwd -apr1 myPassword  
$apr1$qHDFfhPC$nITSVHgYbDAK1Y0acGRnY0
```

## CRYPT

```
openssl passwd -crypt myPassword  
qQ5vTY03c8dsU
```

## Validating CRYPT or MD5 passwords with the OpenSSL command line program

The salt for a CRYPT password is the first two characters (converted to a binary value). To validate myPassword against rqXexS6ZhobKA

## CRYPT

```
$ openssl passwd -crypt -salt rq myPassword  
Warning: truncating password to 8 characters  
rqXexS6ZhobKA
```

Note that using myPasswo instead of myPassword will produce the same result because only the first 8 characters of CRYPT passwords are considered.

The salt for an MD5 password is between \$apr1\$ and the following \$ (as a Base64-encoded binary value - max 8 chars). To validate myPassword against \$apr1\$r31.....\$HqJZimcKQFAMYayBlzkrA/

### MD5

```
$ openssl passwd -apr1 -salt r31..... myPassword  
$apr1$r31.....$HqJZimcKQFAMYayBlzkrA/
```

## Database password fields for mod\_dbd

The SHA1 variant is probably the most useful format for DBD authentication. Since the SHA1 and Base64 functions are commonly available, other software can populate a database with encrypted passwords that are usable by Apache basic authentication.

To create Apache SHA1-variant basic-authentication passwords in various languages:

### PHP

```
'{SHA}' . base64_encode(sha1($password, TRUE))
```

### Java

```
"{SHA}" + new  
sun.misc.BASE64Encoder().encode(java.security.MessageDigest.getIr
```

### ColdFusion

```
"{SHA}" & ToBase64(BinaryDecode(Hash(password, "SHA1"), "Hex"))
```

## Ruby

```
require 'digest/sha1'  
require 'base64'  
'{SHA}' + Base64.encode64(Digest::SHA1.digest(password))
```

## C or C++

Use the APR function: `apr_sha1_base64`

## Python

```
import base64  
import hashlib  
"{SHA}" +  
format(base64.b64encode(hashlib.sha1(password).digest()))
```

## PostgreSQL (with the contrib/pgcrypto functions installed)

```
'{SHA}' || encode(digest(password, 'sha1'), 'base64')
```



Apache recognizes one format for digest-authentication passwords - the MD5 hash of the string `user:realm:password` as a 32-character string of hexadecimal digits. `realm` is the Authorization Realm argument to the [AuthName](#) directive in `httpd.conf`.

## Database password fields for `mod_dbd`

Since the MD5 function is commonly available, other software can populate a database with encrypted passwords that are usable by Apache digest authentication.

To create Apache digest-authentication passwords in various languages:

### PHP

```
md5($user . ':' . $realm . ':' . $password)
```

### Java

```
byte b[] =
java.security.MessageDigest.getInstance("MD5").digest( (user +
":" + realm + ":" + password ).getBytes());
java.math.BigInteger bi = new java.math.BigInteger(1, b);
String s = bi.toString(16);
while (s.length() < 32)
    s = "0" + s;
// String s is the encrypted password
```

### ColdFusion

```
LCASE(Hash( (user & ":" & realm & ":" & password) , "MD5"))
```

### Ruby

```
require 'digest/md5'
Digest::MD5.hexdigest(user + ':' + realm + ':' + password)
```

## PostgreSQL (with the contrib/pgcrypto functions installed)

```
encode(digest( user || ':' || realm || ':' || password ,  
'md5'), 'hex')
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## Shared Object Cache in Apache HTTP Server

The Shared Object Cache provides a means to share simple data across all a server's workers, regardless of [thread and process models](#). It is used where the advantages of sharing data across processes outweigh the performance overhead of inter-process communication.



The shared object cache as such is an abstraction. Four different modules implement it. To use the cache, one or more of these modules must be present, and configured.

The only configuration required is to select which cache provider to use. This is the responsibility of modules using the cache, and they enable selection using directives such as [CacheSocache](#), [AuthnCacheS0Cache](#), [SSLSessionCache](#), and [SSLStaplingCache](#).

Currently available providers are:

**"dbm" ([mod\\_socache\\_dbm](#))**

This makes use of a DBM hash file. The choice of underlying DBM used may be configurable if the installed APR version supports multiple DBM implementations.

**"dc" ([mod\\_socache\\_dc](#))**

This makes use of the [distcache](#) distributed session caching libraries.

**"memcache" ([mod\\_socache\\_memcache](#))**

This makes use of the [memcached](#) high-performance, distributed memory object caching system.

**"shmcb" ([mod\\_socache\\_shmcb](#))**

This makes use of a high-performance cyclic buffer inside a shared memory segment.

The API provides the following functions:

```
const char *create(ap_socache_instance_t **instance, const  
char *arg, apr_pool_t *tmp, apr_pool_t *p);
```

Create a session cache based on the given configuration string. The instance pointer returned in the instance parameter will be passed as the first argument to subsequent

invocations.

**apr\_status\_t init(ap\_socache\_instance\_t \*instance, const char \*cname, const struct ap\_socache\_hints \*hints, server\_rec \*s, apr\_pool\_t \*pool)**

Initialize the cache. The cname must be of maximum length 16 characters, and uniquely identifies the consumer of the cache within the server; using the module name is recommended, e.g. "mod\_ssl-sess". This string may be used within a filesystem path so use of only alphanumeric [a-z0-9\_-] characters is recommended. If hints is non-NULL, it gives a set of hints for the provider. Return APR error code.

**void destroy(ap\_socache\_instance\_t \*instance, server\_rec \*s)**

Destroy a given cache instance object.

**apr\_status\_t store(ap\_socache\_instance\_t \*instance, server\_rec \*s, const unsigned char \*id, unsigned int idlen, apr\_time\_t expiry, unsigned char \*data, unsigned int datalen, apr\_pool\_t \*pool)**

Store an object in a cache instance.

**apr\_status\_t retrieve(ap\_socache\_instance\_t \*instance, server\_rec \*s, const unsigned char \*id, unsigned int idlen, unsigned char \*data, unsigned int \*datalen, apr\_pool\_t \*pool)**

Retrieve a cached object.

**apr\_status\_t remove(ap\_socache\_instance\_t \*instance, server\_rec \*s, const unsigned char \*id, unsigned int idlen, apr\_pool\_t \*pool)**

Remove an object from the cache.

**void status(ap\_socache\_instance\_t \*instance, request\_rec \*r, int flags)**

Dump the status of a cache instance for mod\_status.

**apr\_status\_t iterate(ap\_socache\_instance\_t \*instance, server\_rec \*s, void \*userctx, ap\_socache\_iterator\_t \*iterator,**

**apr\_pool\_t \*pool)**

Dump all cached objects through an iterator callback.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## fcgistarter - Start a FastCGI program



### See also

[mod\\_proxy\\_fcgi](#)



NOTE

Currently only works on Unix systems.



## Synopsis

```
fcgistarter -c command -p port [ -i interface ] -  
N num
```



## Options

- c *command***  
FastCGI program
- p *port***  
Port which the program will listen on
- i *interface***  
Interface which the program will listen on
- N *num***  
Number of instances of the program

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [How-To / Tutorials](#)

## Access Control

Access control refers to any means of controlling access to any resource. This is separate from [authentication and authorization](#).



---

Access control can be done by several different modules. The most important of these are [mod\\_auth\\_core](#) and [mod\\_auth\\_host](#). Also discussed in this document is access control using [mod\\_rewrite](#).



If you wish to restrict access to portions of your site based on the host address of your visitors, this is most easily done using [mod\\_authz\\_host](#).

The [Require](#) provides a variety of different ways to allow or deny access to resources. In conjunction with the [RequireAll](#), [RequireAny](#), and [RequireNone](#) directives, these requirements may be combined in arbitrarily complex ways, to enforce whatever your access policy happens to be.

The [Allow](#), [Deny](#), and [Order](#) directives, provided by [mod\\_access\\_compat](#), are deprecated and will go away in a future version. You should avoid using them, and avoid outdated tutorials recommending their use.

The usage of these directives is:

```
Require host address  
Require ip ip.address
```

In the first form, *address* is a fully qualified domain name (or a partial domain name); you may provide multiple addresses or domain names, if desired.

In the second form, *ip.address* is an IP address, a partial IP address, a network/netmask pair, or a network/nnn CIDR specification. Either IPv4 or IPv6 addresses may be used.

See [the mod\\_authz\\_host documentation](#) for further examples of this syntax.

You can insert `not` to negate a particular requirement. Note, that since a `not` is a negation of a value, it cannot be used by itself to

allow or deny a request, as *not true* does not constitute *false*. Thus, to deny a visit using a negation, the block must have one element that evaluates as true or false. For example, if you have someone spamming your message board, and you want to keep them out, you could do the following:

```
<RequireAll>
  Require all granted
  Require not ip 10.252.46.165
</RequireAll>
```

Visitors coming from that address (10 . 252 . 46 . 165) will not be able to see the content covered by this directive. If, instead, you have a machine name, rather than an IP address, you can use that.

```
Require not host host.example.com
```

And, if you'd like to block access from an entire domain, you can specify just part of an address or domain name:

```
Require not ip 192.168.205
Require not host phishers.example.com moreic
Require not host gov
```

Use of the [RequireAll](#), [RequireAny](#), and [RequireNone](#) directives may be used to enforce more complex sets of requirements.



Using the `<If>`, you can allow or deny access based on arbitrary environment variables or request header values. For example, to deny access based on user-agent (the browser type) you might do the following:

```
<If "%{HTTP_USER_AGENT} == 'BadBot'">  
    Require all denied  
</If>
```

Using the `Require` expr syntax, this could also be written as:

```
Require expr "%{HTTP_USER_AGENT} != 'BadBot'
```

**Warning:**

Access control by User-Agent is an unreliable technique, since the User-Agent header can be set to anything at all, at the whim of the end user.

See [the expressions document](#) for a further discussion of what expression syntaxes and variables are available to you.



The [F] [RewriteRule](#) flag causes a 403 Forbidden response to be sent. Using this, you can deny access to a resource based on arbitrary criteria.

For example, if you wish to block access to a resource between 8pm and 7am, you can do this using [mod\\_rewrite](#).

```
RewriteEngine On
RewriteCond "%{TIME_HOUR}" ">=20" [OR]
RewriteCond "%{TIME_HOUR}" "<07"
RewriteRule "^/fridge" "-" [F]
```

This will return a 403 Forbidden response for any request after 8pm or before 7am. This technique can be used for any criteria that you wish to check. You can also redirect, or otherwise rewrite these requests, if that approach is preferred.

The [<If>](#) directive, added in 2.4, replaces many things that [mod\\_rewrite](#) has traditionally been used to do, and you should probably look there first before resorting to `mod_rewrite`.



The [expression engine](#) gives you a great deal of power to do a variety of things based on arbitrary server variables, and you should consult that document for more detail.

Also, you should read the [mod\\_auth\\_core](#) documentation for examples of combining multiple access requirements and specifying how they interact.

See also the [Authentication and Authorization](#) howto.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

# Caching Guide

This document supplements the [mod\\_cache](#), [mod\\_cache\\_disk](#), [mod\\_file\\_cache](#) and [htcacheclean](#) reference documentation. It describes how to use the Apache HTTP Server's caching features to accelerate web and proxy serving, while avoiding common problems and misconfigurations.



---

The Apache HTTP server offers a range of caching features that are designed to improve the performance of the server in various ways.

### **Three-state RFC2616 HTTP caching**

[mod\\_cache](#) and its provider modules [mod\\_cache\\_disk](#) provide intelligent, HTTP-aware caching. The content itself is stored in the cache, and `mod_cache` aims to honor all of the various HTTP headers and options that control the cacheability of content as described in [Section 13 of RFC2616](#). [mod\\_cache](#) is aimed at both simple and complex caching configurations, where you are dealing with proxied content, dynamic local content or have a need to speed up access to local files on a potentially slow disk.

### **Two-state key/value shared object caching**

The [shared object cache API](#) (socache) and its provider modules provide a server wide key/value based shared object cache. These modules are designed to cache low level data such as SSL sessions and authentication credentials. Backends allow the data to be stored server wide in shared memory, or datacenter wide in a cache such as memcache or distcache.

### **Specialized file caching**

[mod\\_file\\_cache](#) offers the ability to pre-load files into memory on server startup, and can improve access times and save file handles on files that are accessed often, as there is no need to go to disk on each request.

To get the most from this document, you should be familiar with the basics of HTTP, and have read the Users' Guides to [Mapping URLs to the Filesystem](#) and [Content negotiation](#).



## Three States of an HTTP Cache

Related Modules	Related Directives
<a href="#">mod_cache</a>	<a href="#">CacheEnable</a>
<a href="#">mod_cache_disk</a>	<a href="#">CacheDisable</a>
	<a href="#">UseCanonicalName</a>
	<a href="#">CacheNegotiatedDocs</a>

The HTTP protocol contains built in support for an in-line caching mechanism [described by section 13 of RFC2616](#), and the [mod\\_cache](#) module can be used to take advantage of this.

Unlike a simple two state key/value cache where the content disappears completely when no longer fresh, an HTTP cache includes a mechanism to retain stale content, and to ask the origin server whether this stale content has changed and if not, make it fresh again.

An entry in an HTTP cache exists in one of three states:

### Fresh

If the content is new enough (younger than its **freshness lifetime**), it is considered **fresh**. An HTTP cache is free to serve fresh content without making any calls to the origin server at all.

### Stale

If the content is too old (older than its **freshness lifetime**), it is considered **stale**. An HTTP cache should contact the origin server and check whether the content is still fresh before serving stale content to a client. The origin server will either respond with replacement content if not still valid, or ideally, the origin server will respond with a code to tell the cache the content is still fresh, without the need to generate or send the content again. The content becomes fresh again and the cycle continues.

The HTTP protocol does allow the cache to serve stale data under certain circumstances, such as when an attempt to freshen the data with an origin server has failed with a 5xx error, or when another request is already in the process of freshening the given entry. In these cases a Warning header is added to the response.

## Non Existent

If the cache gets full, it reserves the option to delete content from the cache to make space. Content can be deleted at any time, and can be stale or fresh. The [htcacheclean](#) tool can be run on a once off basis, or deployed as a daemon to keep the size of the cache within the given size, or the given number of inodes. The tool attempts to delete stale content before attempting to delete fresh content.

Full details of how HTTP caching works can be found in [Section 13 of RFC2616](#).

## Interaction with the Server

The [mod\\_cache](#) module hooks into the server in two possible places depending on the value of the [CacheQuickHandler](#) directive:

### Quick handler phase

This phase happens very early on during the request processing, just after the request has been parsed. If the content is found within the cache, it is served immediately and almost all request processing is bypassed.

In this scenario, the cache behaves as if it has been "bolted on" to the front of the server.

This mode offers the best performance, as the majority of server processing is bypassed. This mode however also

bypasses the authentication and authorization phases of server processing, so this mode should be chosen with care when this is important.

Requests with an "Authorization" header (for example, HTTP Basic Authentication) are neither cacheable nor served from the cache when [mod\\_cache](#) is running in this phase.

### **Normal handler phase**

This phase happens late in the request processing, after all the request phases have completed.

In this scenario, the cache behaves as if it has been "bolted on" to the back of the server.

This mode offers the most flexibility, as the potential exists for caching to occur at a precisely controlled point in the filter chain, and cached content can be filtered or personalized before being sent to the client.

If the URL is not found within the cache, [mod\\_cache](#) will add a [filter](#) to the filter stack in order to record the response to the cache, and then stand down, allowing normal request processing to continue. If the content is determined to be cacheable, the content will be saved to the cache for future serving, otherwise the content will be ignored.

If the content found within the cache is stale, the [mod\\_cache](#) module converts the request into a **conditional request**. If the origin server responds with a normal response, the normal response is cached, replacing the content already cached. If the origin server responds with a 304 Not Modified response, the content is marked as fresh again, and the cached content is served by the filter instead of saving it.

## Improving Cache Hits

When a virtual host is known by one of many different server aliases, ensuring that [UseCanonicalName](#) is set to On can dramatically improve the ratio of cache hits. This is because the hostname of the virtual-host serving the content is used within the cache key. With the setting set to On virtual-hosts with multiple server names or aliases will not produce differently cached entities, and instead content will be cached as per the canonical hostname.

## Freshness Lifetime

Well formed content that is intended to be cached should declare an explicit freshness lifetime with the `Cache-Control` header's `max-age` or `s-maxage` fields, or by including an `Expires` header.

At the same time, the origin server defined freshness lifetime can be overridden by a client when the client presents their own `Cache-Control` header within the request. In this case, the lowest freshness lifetime between request and response wins.

When this freshness lifetime is missing from the request or the response, a default freshness lifetime is applied. The default freshness lifetime for cached entities is one hour, however this can be easily over-ridden by using the [CacheDefaultExpire](#) directive.

If a response does not include an `Expires` header but does include a `Last-Modified` header, `mod_cache` can infer a freshness lifetime based on a heuristic, which can be controlled through the use of the [CacheLastModifiedFactor](#) directive.

For local content, or for remote content that does not define its

own Expires header, [mod\\_expires](#) may be used to fine-tune the freshness lifetime by adding max-age and Expires.

The maximum freshness lifetime may also be controlled by using the [CacheMaxExpire](#).

## A Brief Guide to Conditional Requests

When content expires from the cache and becomes stale, rather than pass on the original request, httpd will modify the request to make it conditional instead.

When an ETag header exists in the original cached response, [mod\\_cache](#) will add an If-None-Match header to the request to the origin server. When a Last-Modified header exists in the original cached response, [mod\\_cache](#) will add an If-Modified-Since header to the request to the origin server. Performing either of these actions makes the request **conditional**.

When a conditional request is received by an origin server, the origin server should check whether the ETag or the Last-Modified parameter has changed, as appropriate for the request. If not, the origin should respond with a terse "304 Not Modified" response. This signals to the cache that the stale content is still fresh should be used for subsequent requests until the content's new freshness lifetime is reached again.

If the content has changed, then the content is served as if the request were not conditional to begin with.

Conditional requests offer two benefits. Firstly, when making such a request to the origin server, if the content from the origin matches the content in the cache, this can be determined easily and without the overhead of transferring the entire resource.

Secondly, a well designed origin server will be designed in such a way that conditional requests will be significantly cheaper to produce than a full response. For static files, typically all that is involved is a call to `stat ( )` or similar system call, to see if the file has changed in size or modification time. As such, even local content may still be served faster from the cache if it has not changed.

Origin servers should make every effort to support conditional requests as is practical, however if conditional requests are not supported, the origin will respond as if the request was not conditional, and the cache will respond as if the content had changed and save the new content to the cache. In this case, the cache will behave like a simple two state cache, where content is effectively either fresh or deleted.

## What Can be Cached?

The full definition of which responses can be cached by an HTTP cache is defined in [RFC2616 Section 13.4 Response Cacheability](#), and can be summed up as follows:

1. Caching must be enabled for this URL. See the [CacheEnable](#) and [CacheDisable](#) directives.
2. The response must have a HTTP status code of 200, 203, 300, 301 or 410.
3. The request must be a HTTP GET request.
4. If the response contains an "Authorization:" header, it must also contain an "s-maxage", "must-revalidate" or "public" option in the "Cache-Control:" header, or it won't be cached.
5. If the URL included a query string (e.g. from a HTML form GET method) it will not be cached unless the response specifies an explicit expiration by including an "Expires:"

header or the max-age or s-maxage directive of the "Cache-Control:" header, as per RFC2616 sections 13.9 and 13.2.1.

6. If the response has a status of 200 (OK), the response must also include at least one of the "Etag", "Last-Modified" or the "Expires" headers, or the max-age or s-maxage directive of the "Cache-Control:" header, unless the [CacheIgnoreNoLastMod](#) directive has been used to require otherwise.
7. If the response includes the "private" option in a "Cache-Control:" header, it will not be stored unless the [CacheStorePrivate](#) has been used to require otherwise.
8. Likewise, if the response includes the "no-store" option in a "Cache-Control:" header, it will not be stored unless the [CacheStoreNoStore](#) has been used.
9. A response will not be stored if it includes a "Vary:" header containing the match-all "\*".

## What Should Not be Cached?

It should be up to the client creating the request, or the origin server constructing the response to decide whether or not the content should be cacheable or not by correctly setting the Cache-Control header, and [mod\\_cache](#) should be left alone to honor the wishes of the client or server as appropriate.

Content that is time sensitive, or which varies depending on the particulars of the request that are not covered by HTTP negotiation, should not be cached. This content should declare itself uncacheable using the Cache-Control header.

If content changes often, expressed by a freshness lifetime of minutes or seconds, the content can still be cached, however it is highly desirable that the origin server supports **conditional**

**requests** correctly to ensure that full responses do not have to be generated on a regular basis.

Content that varies based on client provided request headers can be cached through intelligent use of the Vary response header.

## Variable/Negotiated Content

When the origin server is designed to respond with different content based on the value of headers in the request, for example to serve multiple languages at the same URL, HTTP's caching mechanism makes it possible to cache multiple variants of the same page at the same URL.

This is done by the origin server adding a Vary header to indicate which headers must be taken into account by a cache when determining whether two variants are different from one another.

If for example, a response is received with a vary header such as;

```
Vary: negotiate, accept-language, accept-charset
```

[mod\\_cache](#) will only serve the cached content to requesters with accept-language and accept-charset headers matching those of the original request.

Multiple variants of the content can be cached side by side, [mod\\_cache](#) uses the Vary header and the corresponding values of the request headers listed by Vary to decide on which of many variants to return to the client.



Related Modules	Related Directives
<a href="#">mod_cache</a>	<a href="#">CacheEnable</a>
<a href="#">mod_cache_disk</a>	<a href="#">CacheRoot</a>
<a href="#">mod_cache_socache</a>	<a href="#">CacheDirLevels</a>
<a href="#">mod_socache_memcache</a>	<a href="#">CacheDirLength</a>
	<a href="#">CacheSocache</a>

## Caching to Disk

The [mod\\_cache](#) module relies on specific backend store implementations in order to manage the cache, and for caching to disk [mod\\_cache\\_disk](#) is provided to support this.

Typically the module will be configured as so;

```
CacheRoot    "/var/cache/apache/"
CacheEnable  disk /
CacheDirLevels 2
CacheDirLength 1
```

Importantly, as the cached files are locally stored, operating system in-memory caching will typically be applied to their access also. So although the files are stored on disk, if they are frequently accessed it is likely the operating system will ensure that they are actually served from memory.

## Understanding the Cache-Store

To store items in the cache, [mod\\_cache\\_disk](#) creates a 22 character hash of the URL being requested. This hash incorporates the hostname, protocol, port, path and any CGI arguments to the URL, as well as elements defined by the Vary header to ensure that multiple URLs do not collide with one

another.

Each character may be any one of 64-different characters, which mean that overall there are  $64^{22}$  possible hashes. For example, a URL might be hashed to xyTGxSM02b68mBCykqkp1w. This hash is used as a prefix for the naming of the files specific to that URL within the cache, however first it is split up into directories as per the [CacheDirLevels](#) and [CacheDirLength](#) directives.

[CacheDirLevels](#) specifies how many levels of subdirectory there should be, and [CacheDirLength](#) specifies how many characters should be in each directory. With the example settings given above, the hash would be turned into a filename prefix as /var/cache/apache/x/y/TGxSM02b68mBCykqkp1w.

The overall aim of this technique is to reduce the number of subdirectories or files that may be in a particular directory, as most file-systems slow down as this number increases. With setting of "1" for [CacheDirLength](#) there can at most be 64 subdirectories at any particular level. With a setting of 2 there can be  $64 * 64$  subdirectories, and so on. Unless you have a good reason not to, using a setting of "1" for [CacheDirLength](#) is recommended.

Setting [CacheDirLevels](#) depends on how many files you anticipate to store in the cache. With the setting of "2" used in the above example, a grand total of 4096 subdirectories can ultimately be created. With 1 million files cached, this works out at roughly 245 cached URLs per directory.

Each URL uses at least two files in the cache-store. Typically there is a ".header" file, which includes meta-information about the URL, such as when it is due to expire and a ".data" file which is a verbatim copy of the content to be served.

In the case of a content negotiated via the "Vary" header, a ".vary"

directory will be created for the URL in question. This directory will have multiple ".data" files corresponding to the differently negotiated content.

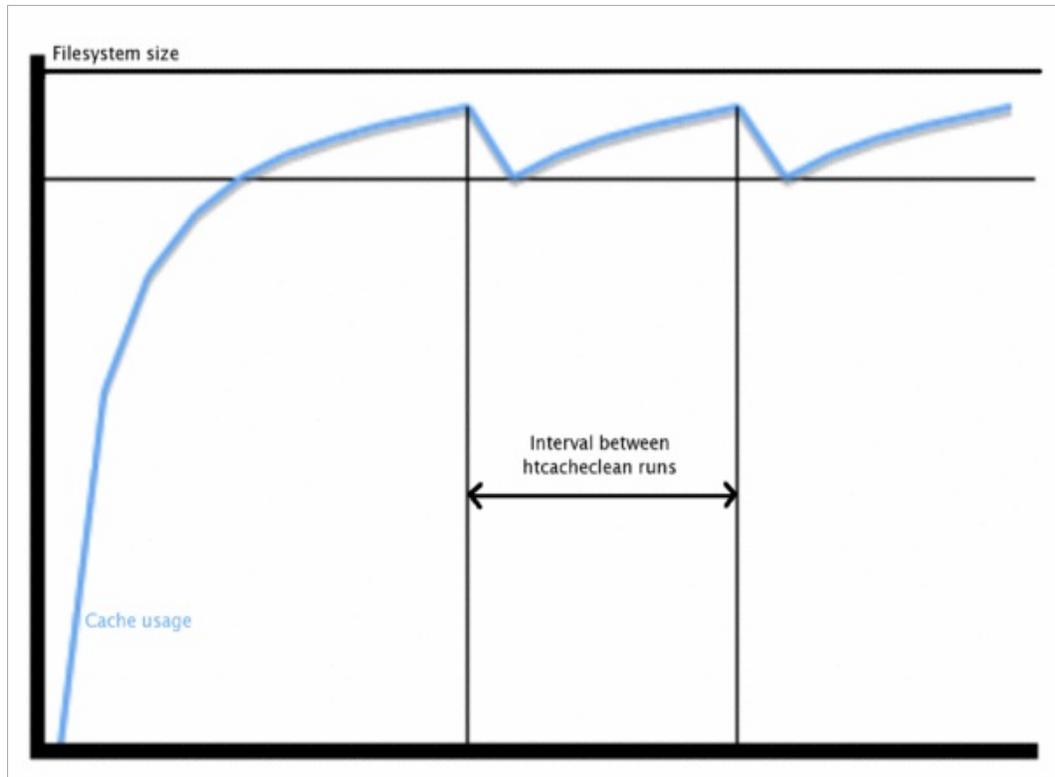
## Maintaining the Disk Cache

The `mod_cache_disk` module makes no attempt to regulate the amount of disk space used by the cache, although it will gracefully stand down on any disk error and behave as if the cache was never present.

Instead, provided with httpd is the `htcacheclean` tool which allows you to clean the cache periodically. Determining how frequently to run `htcacheclean` and what target size to use for the cache is somewhat complex and trial and error may be needed to select optimal values.

`htcacheclean` has two modes of operation. It can be run as persistent daemon, or periodically from cron. `htcacheclean` can take up to an hour or more to process very large (tens of gigabytes) caches and if you are running it from cron it is recommended that you determine how long a typical run takes, to avoid running more than one instance at a time.

It is also recommended that an appropriate "nice" level is chosen for `htcacheclean` so that the tool does not cause excessive disk io while the server is running.



**Figure 1:** Typical cache growth / clean sequence.

Because `mod_cache_disk` does not itself pay attention to how much space is used you should ensure that `htcacheclean` is configured to leave enough "grow room" following a clean.

## Caching to memcached

Using the `mod_cache_socache` module, `mod_cache` can cache data from a variety of implementations (aka: "providers"). Using the `mod_socache_memcache` module, for example, one can specify that `memcached` is to be used as the the backend storage mechanism.

Typically the module will be configured as so:

```
CacheEnable socache /
CacheSocache memcache:memcd.example.com:1121
```

Additional memcached servers can be specified by appending them to the end of the CacheSocache memcache: line separated by commas:

```
CacheEnable socache /  
CacheSocache memcache:mem1.example.com:11211
```

This format is also used with the other various [mod\\_cache\\_socache](#) providers. For example:

```
CacheEnable socache /  
CacheSocache shmcb:/path/to/datafile(512000)
```

```
CacheEnable socache /  
CacheSocache dbm:/path/to/datafile
```



Related Modules	Related Directives
<a href="#">mod_authn_socache</a>	<a href="#">AuthnCacheSocache</a>
<a href="#">mod_socache_dbm</a>	<a href="#">SSLSessionCache</a>
<a href="#">mod_socache_dc</a>	<a href="#">SSLStaplingCache</a>
<a href="#">mod_socache_memcache</a>	
<a href="#">mod_socache_shmcb</a>	
<a href="#">mod_ssl</a>	

The Apache HTTP server offers a low level shared object cache for caching information such as SSL sessions, or authentication credentials, within the [socache](#) interface.

Additional modules are provided for each implementation, offering the following backends:

**[mod\\_socache\\_dbm](#)**

DBM based shared object cache.

**[mod\\_socache\\_dc](#)**

Distcache based shared object cache.

**[mod\\_socache\\_memcache](#)**

Memcache based shared object cache.

**[mod\\_socache\\_shmcb](#)**

Shared memory based shared object cache.

**Caching Authentication Credentials**

Related Modules	Related Directives
<a href="#">mod_authn_socache</a>	<a href="#">AuthnCacheSocache</a>

The [mod\\_authn\\_socache](#) module allows the result of authentication to be cached, relieving load on authentication

backends.

## Caching SSL Sessions

Related Modules	Related Directives
<a href="#">mod_ssl</a>	<a href="#">SSLSessionCache</a> <a href="#">SSLStaplingCache</a>

The [mod\\_ssl](#) module uses the socache interface to provide a session cache and a stapling cache.



## File-Handle Caching

Related Modules	Related Directives
<a href="#">mod_file_cache</a>	<a href="#">CacheFile</a>
	<a href="#">MMapFile</a>

On platforms where a filesystem might be slow, or where file handles are expensive, the option exists to pre-load files into memory on startup.

On systems where opening files is slow, the option exists to open the file on startup and cache the file handle. These options can help on systems where access to static files is slow.

### File-Handle Caching

The act of opening a file can itself be a source of delay, particularly on network filesystems. By maintaining a cache of open file descriptors for commonly served files, httpd can avoid this delay. Currently httpd provides one implementation of File-Handle Caching.

#### CacheFile

The most basic form of caching present in httpd is the file-handle caching provided by [mod\\_file\\_cache](#). Rather than caching file-contents, this cache maintains a table of open file descriptors. Files to be cached in this manner are specified in the configuration file using the [CacheFile](#) directive.

The [CacheFile](#) directive instructs httpd to open the file when it is started and to re-use this file-handle for all subsequent access to this file.

```
CacheFile /usr/local/apache2/htdocs/index.html
```

If you intend to cache a large number of files in this manner, you must ensure that your operating system's limit for the number of open files is set appropriately.

Although using `CacheFile` does not cause the file-contents to be cached per-se, it does mean that if the file changes while httpd is running these changes will not be picked up. The file will be consistently served as it was when httpd was started.

If the file is removed while httpd is running, it will continue to maintain an open file descriptor and serve the file as it was when httpd was started. This usually also means that although the file will have been deleted, and not show up on the filesystem, extra free space will not be recovered until httpd is stopped and the file descriptor closed.

## **In-Memory Caching**

Serving directly from system memory is universally the fastest method of serving content. Reading files from a disk controller or, even worse, from a remote network is orders of magnitude slower. Disk controllers usually involve physical processes, and network access is limited by your available bandwidth. Memory access on the other hand can take mere nano-seconds.

System memory isn't cheap though, byte for byte it's by far the most expensive type of storage and it's important to ensure that it is used efficiently. By caching files in memory you decrease the amount of memory available on the system. As we'll see, in the case of operating system caching, this is not so much of an issue, but when using httpd's own in-memory caching it is important to make sure that you do not allocate too much memory to a cache. Otherwise the system will be forced to swap out memory, which will likely degrade performance.

## Operating System Caching

Almost all modern operating systems cache file-data in memory managed directly by the kernel. This is a powerful feature, and for the most part operating systems get it right. For example, on Linux, let's look at the difference in the time it takes to read a file for the first time and the second time;

```
colm@coroebus:~$ time cat testfile > /dev/null
real    0m0.065s
user    0m0.000s
sys     0m0.001s
colm@coroebus:~$ time cat testfile > /dev/null
real    0m0.003s
user    0m0.003s
sys     0m0.000s
```

Even for this small file, there is a huge difference in the amount of time it takes to read the file. This is because the kernel has cached the file contents in memory.

By ensuring there is "spare" memory on your system, you can ensure that more and more file-contents will be stored in this cache. This can be a very efficient means of in-memory caching, and involves no extra configuration of httpd at all.

Additionally, because the operating system knows when files are deleted or modified, it can automatically remove file contents from the cache when necessary. This is a big advantage over httpd's in-memory caching which has no way of knowing when a file has changed.

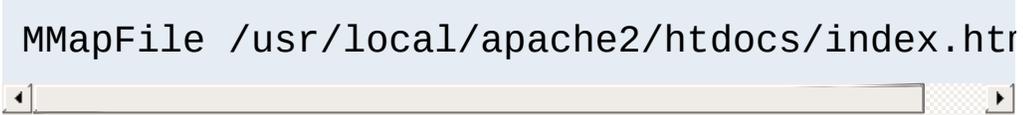
Despite the performance and advantages of automatic operating system caching there are some circumstances in which in-memory caching may be better performed by httpd.

## MMapFile Caching

[mod\\_file\\_cache](#) provides the [MMapFile](#) directive, which allows

you to have httpd map a static file's contents into memory at start time (using the mmap system call). httpd will use the in-memory contents for all subsequent accesses to this file.

```
MMapFile /usr/local/apache2/htdocs/index.htm
```



As with the [CacheFile](#) directive, any changes in these files will not be picked up by httpd after it has started.

The [MMapFile](#) directive does not keep track of how much memory it allocates, so you must ensure not to over-use the directive. Each httpd child process will replicate this memory, so it is critically important to ensure that the files mapped are not so large as to cause the system to swap memory.



## Authorization and Access Control

Using `mod_cache` in its default state where `CacheQuickHandler` is set to `On` is very much like having a caching reverse-proxy bolted to the front of the server. Requests will be served by the caching module unless it determines that the origin server should be queried just as an external cache would, and this drastically changes the security model of `httpd`.

As traversing a filesystem hierarchy to examine potential `.htaccess` files would be a very expensive operation, partially defeating the point of caching (to speed up requests), `mod_cache` makes no decision about whether a cached entity is authorised for serving. In other words; if `mod_cache` has cached some content, it will be served from the cache as long as that content has not expired.

If, for example, your configuration permits access to a resource by IP address you should ensure that this content is not cached. You can do this by using the `CacheDisable` directive, or `mod_expires`. Left unchecked, `mod_cache` - very much like a reverse proxy - would cache the content when served and then serve it to any client, on any IP address.

When the `CacheQuickHandler` directive is set to `Off`, the full set of request processing phases are executed and the security model remains unchanged.

## Local exploits

As requests to end-users can be served from the cache, the cache itself can become a target for those wishing to deface or interfere with content. It is important to bear in mind that the cache must at all times be writable by the user which `httpd` is running as. This is

in stark contrast to the usually recommended situation of maintaining all content unwritable by the Apache user.

If the Apache user is compromised, for example through a flaw in a CGI process, it is possible that the cache may be targeted. When using [mod\\_cache\\_disk](#), it is relatively easy to insert or modify a cached entity.

This presents a somewhat elevated risk in comparison to the other types of attack it is possible to make as the Apache user. If you are using [mod\\_cache\\_disk](#) you should bear this in mind - ensure you upgrade httpd when security upgrades are announced and run CGI processes as a non-Apache user using [suEXEC](#) if possible.

## Cache Poisoning

When running httpd as a caching proxy server, there is also the potential for so-called cache poisoning. Cache Poisoning is a broad term for attacks in which an attacker causes the proxy server to retrieve incorrect (and usually undesirable) content from the origin server.

For example if the DNS servers used by your system running httpd are vulnerable to DNS cache poisoning, an attacker may be able to control where httpd connects to when requesting content from the origin server. Another example is so-called HTTP request-smuggling attacks.

This document is not the correct place for an in-depth discussion of HTTP request smuggling (instead, try your favourite search engine) however it is important to be aware that it is possible to make a series of requests, and to exploit a vulnerability on an origin webserver such that the attacker can entirely control the content retrieved by the proxy.

## Denial of Service / Cachebusting

The Vary mechanism allows multiple variants of the same URL to be cached side by side. Depending on header values provided by the client, the cache will select the correct variant to return to the client. This mechanism can become a problem when an attempt is made to vary on a header that is known to contain a wide range of possible values under normal use, for example the User-Agent header. Depending on the popularity of the particular web site thousands or millions of duplicate cache entries could be created for the same URL, crowding out other entries in the cache.

In other cases, there may be a need to change the URL of a particular resource on every request, usually by adding a "cachebuster" string to the URL. If this content is declared cacheable by a server for a significant freshness lifetime, these entries can crowd out legitimate entries in a cache. While [mod\\_cache](#) provides a [CacheIgnoreURLSessionIdentifiers](#) directive, this directive should be used with care to ensure that downstream proxy or browser caches aren't subjected to the same denial of service issue.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## httxt2dbm - Generate dbm files for use with RewriteMap

httxt2dbm is used to generate dbm files from text input, for use in [RewriteMap](#) with the dbm map type.

If the output file already exists, it will not be truncated. New keys will be added and existing keys will be updated.



### See also

[httpd](#)

[mod\\_rewrite](#)



## Synopsis

```
httxt2dbm [ -v ] [ -f DBM_TYPE ] -i SOURCE_TXT -o  
OUTPUT_DBM
```



**-v**

More verbose output

**-f *DBM\_TYPE***

Specify the DBM type to be used for the output. If not specified, will use the [APR](#) Default. Available types are: GDBM for GDBM files, SDBM for SDBM files, DB for Berkeley DB files, NDBM for NDBM files, default for the default DBM type.

**-i *SOURCE\_TXT***

Input file from which the dbm is to be created. The file should be formatted with one record per line, of the form: key value. See the documentation for [RewriteMap](#) for further details of this file's format and meaning.

**-o *OUTPUT\_DBM***

Name of the output dbm files.



## Examples

```
httxt2dbm -i rewritermap.txt -o rewritermap.dbm  
httxt2dbm -f SDBM -i rewritermap.txt -o rewritermap.dbm
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Using RewriteMap

This document supplements the [mod\\_rewrite reference documentation](#). It describes the use of the [RewriteMap](#) directive, and provides examples of each of the various [RewriteMap](#) types.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.



## See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Redirection and remapping](#)
- [Controlling access](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



The [RewriteMap](#) directive defines an external function which can be called in the context of [RewriteRule](#) or [RewriteCond](#) directives to perform rewriting that is too complicated, or too specialized to be performed just by regular expressions. The source of this lookup can be any of the types listed in the sections below, and enumerated in the [RewriteMap](#) reference documentation.

The syntax of the [RewriteMap](#) directive is as follows:

```
RewriteMap MapName MapType:MapSource
```

The *MapName* is an arbitrary name that you assign to the map, and which you will use in directives later on. Arguments are passed to the map via the following syntax:

```
#{ MapName : LookupKey } #{ MapName : LookupKey |  
DefaultValue }
```

When such a construct occurs, the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the key is not found then it is substituted by *DefaultValue* or by the empty string if no *DefaultValue* was specified.

For example, you can define a [RewriteMap](#) as:

```
RewriteMap examplemap "txt:/path/to/file/map"
```

You would then be able to use this map in a [RewriteRule](#) as follows:

```
RewriteRule "^/ex/(.*)" "${examplemap:$1}"
```

A default value can be specified in the event that nothing is found in the map:

```
RewriteRule "^/ex/(.*)" "${examplemap:$1|/no
```

### Per-directory and .htaccess context

The [RewriteMap](#) directive may not be used in [<Directory>](#) sections or `.htaccess` files. You must declare the map in server or virtualhost context. You may use the map, once created, in your [RewriteRule](#) and [RewriteCond](#) directives in those scopes. You just can't **declare** it in those scopes.

The sections that follow describe the various *MapTypes* that may be used, and give examples of each.



When a `MapType` of `int` is used, the `MapSource` is one of the available internal [RewriteMap](#) functions. Module authors can provide additional internal functions by registering them with the `ap_register_rewrite_mapfunc` API. The functions that are provided by default are:

- **toupper:**  
Converts the key to all upper case.
- **tolower:**  
Converts the key to all lower case.
- **escape:**  
Translates special characters in the key to hex-encodings.
- **unescape:**  
Translates hex-encodings in the key back to special characters.

To use one of these functions, create a [RewriteMap](#) referencing the `int` function, and then use that in your [RewriteRule](#):

### Redirect a URI to an all-lowercase version of itself

```
RewriteMap lc int:tolower
RewriteRule "(.*)" "${lc:$1}" [R]
```

Please note that the example offered here is for illustration purposes only, and is not a recommendation. If you want to make URLs case-insensitive, consider using [mod\\_speling](#) instead.



## TEXT MAPS

When a MapType of txt is used, the MapSource is a filesystem path to a plain-text mapping file, containing one space-separated key/value pair per line. Optionally, a line may contain a comment, starting with a '#' character.

A valid text rewrite map file will have the following syntax:

```
# Comment line
MatchingKey SubstValue
MatchingKey SubstValue # comment
```

When the [RewriteMap](#) is invoked the argument is looked for in the first argument of a line, and, if found, the substitution value is returned.

For example, we can use a mapfile to translate product names to product IDs for easier-to-remember URLs, using the following recipe:

### Product to ID configuration

```
RewriteMap product2id "txt:/etc/apache2/prod
RewriteRule "^/product/(.*)" "/prods.php?id=
```

We assume here that the prods.php script knows what to do when it received an argument of id=NOTFOUND when a product is not found in the lookup map.

The file /etc/apache2/productmap.txt then contains the following:

### Product to ID map

```
##
## productmap.txt - Product to ID map file
```

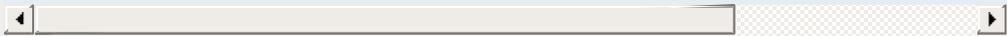
```
##  
  
television 993  
stereo 198  
fishingrod 043  
basketball 418  
telephone 328
```

Thus, when `http://example.com/product/television` is requested, the [RewriteRule](#) is applied, and the request is internally mapped to `/prods.php?id=993`.

### Note: .htaccess files

The example given is crafted to be used in server or virtualhost scope. If you're planning to use this in a `.htaccess` file, you'll need to remove the leading slash from the rewrite pattern in order for it to match anything:

```
RewriteRule "^product/(.*)" "/prods.php?id=${product2id:$1
```



### Cached lookups

The looked-up keys are cached by `httpd` until the `mtime` (modified time) of the mapfile changes, or the `httpd` server is restarted. This ensures better performance on maps that are called by many requests.



When a MapType of rnd is used, the MapSource is a filesystem path to a plain-text mapping file, each line of which contains a key, and one or more values separated by |. One of these values will be chosen at random if the key is matched.

For example, you can use the following map file and directives to provide a random load balancing between several back-end servers, via a reverse-proxy. Images are sent to one of the servers in the 'static' pool, while everything else is sent to one of the 'dynamic' pool.

### Rewrite map file

```
##  
## map.txt -- rewriting map  
##  
  
static www1|www2|www3|www4  
dynamic www5|www6
```

### Configuration directives

```
RewriteMap servers "rnd:/path/to/file/map.txt"  
  
RewriteRule "^/(.*\.(png|gif|jpg))" "http://  
RewriteRule "^/(.*)" "http://
```

So, when an image is requested and the first of these rules is matched, [RewriteMap](#) looks up the string `static` in the map file, which returns one of the specified hostnames at random, which is then used in the [RewriteRule](#) target.

If you wanted to have one of the servers more likely to be chosen (for example, if one of the server has more memory than the others, and so can handle more requests) simply list it more times

in the map file.

```
static www1|www1|www2|www3|www4
```



When a MapType of dbm is used, the MapSource is a filesystem path to a DBM database file containing key/value pairs to be used in the mapping. This works exactly the same way as the txt map, but is much faster, because a DBM is indexed, whereas a text file is not. This allows more rapid access to the desired key.

You may optionally specify a particular dbm type:

```
RewriteMap examplemap "dbm=sdbm:/etc/apache/
```

The type can be sdbm, gdbm, ndbm or db. However, it is recommended that you just use the [httxt2dbm](#) utility that is provided with Apache HTTP Server, as it will use the correct DBM library, matching the one that was used when httpd itself was built.

To create a dbm file, first create a text map file as described in the [txt](#) section. Then run `httxt2dbm`:

```
$ htxt2dbm -i mapfile.txt -o mapfile.map
```

You can then reference the resulting file in your [RewriteMap](#) directive:

```
RewriteMap mapname "dbm:/etc/apache/mapfile
```

Note that with some dbm types, more than one file is generated, with a common base name. For example, you may have two files named `mapfile.map.dir` and `mapfile.map.pag`. This is normal, and you need only use the base name `mapfile.map` in your [RewriteMap](#) directive.

## Cached lookups

The looked-up keys are cached by httpd until the `mtime` (modified time) of the mapfile changes, or the httpd server is restarted. This ensures better performance on maps that are called by many requests.



## prg External Rewriting Program

When a MapType of prg is used, the MapSource is a filesystem path to an executable program which will providing the mapping behavior. This can be a compiled binary file, or a program in an interpreted language such as Perl or Python.

This program is started once, when the Apache HTTP Server is started, and then communicates with the rewriting engine via STDIN and STDOUT. That is, for each map function lookup, it expects one argument via STDIN, and should return one new-line terminated response string on STDOUT. If there is no corresponding lookup value, the map program should return the four-character string "NULL" to indicate this.

External rewriting programs are not started if they're defined in a context that does not have [RewriteEngine](#) set to on.

This feature utilizes the `rewrite-map` mutex, which is required for reliable communication with the program. The mutex mechanism and lock file can be configured with the [Mutex](#) directive.

A simple example is shown here which will replace all dashes with underscores in a request URI.

### Rewrite configuration

```
RewriteMap d2u "prg:/www/bin/dash2under.pl"  
RewriteRule "-" "${d2u:%{REQUEST_URI}}"
```

### dash2under.pl

```
#!/usr/bin/perl  
$| = 1; # Turn off I/O buffering
```

```
while (<STDIN>) {  
    s/-/_/g; # Replace dashes with underscore  
    print $_;  
}
```

### Caution!

- Keep your rewrite map program as simple as possible. If the program hangs, it will cause httpd to wait indefinitely for a response from the map, which will, in turn, cause httpd to stop responding to requests.
- Be sure to turn off buffering in your program. In Perl this is done by the second line in the example script: `$| = 1;` This will of course vary in other languages. Buffered I/O will cause httpd to wait for the output, and so it will hang.
- Remember that there is only one copy of the program, started at server startup. All requests will need to go through this one bottleneck. This can cause significant slowdowns if many requests must go through this process, or if the script itself is very slow.



When a MapType of `dbd` or `fastdbd` is used, the MapSource is a SQL `SELECT` statement that takes a single argument and returns a single value.

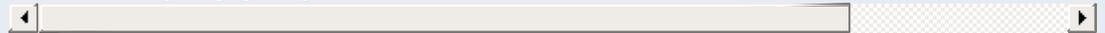
`mod_dbd` will need to be configured to point at the right database for this statement to be executed.

There are two forms of this MapType. Using a MapType of `dbd` causes the query to be executed with each map request, while using `fastdbd` caches the database lookups internally. So, while `fastdbd` is more efficient, and therefore faster, it won't pick up on changes to the database until the server is restarted.

If a query returns more than one row, a random row from the result set is used.

### Example

```
RewriteMap myquery "fastdbd:SELECT destination FROM rewrite WHEI
```



The [RewriteMap](#) directive can occur more than once. For each mapping-function use one [RewriteMap](#) directive to declare its rewriting mapfile.

While you cannot **declare** a map in per-directory context (`.htaccess` files or [<Directory>](#) blocks) it is possible to **use** this map in per-directory context.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Apache mod\_rewrite Introduction

This document supplements the [mod\\_rewrite reference documentation](#). It describes the basic concepts necessary for use of [mod\\_rewrite](#). Other documents go into greater detail, but this doc should help the beginner get their feet wet.



## See also

- [Module documentation](#)
- [Redirection and remapping](#)
- [Controlling access](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Using RewriteMap](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



---

The Apache module [mod\\_rewrite](#) is a very powerful and sophisticated module which provides a way to do URL manipulations. With it, you can do nearly all types of URL rewriting that you may need. It is, however, somewhat complex, and may be intimidating to the beginner. There is also a tendency to treat rewrite rules as magic incantation, using them without actually understanding what they do.

This document attempts to give sufficient background so that what follows is understood, rather than just copied blindly.

Remember that many common URL-manipulation tasks don't require the full power and complexity of [mod\\_rewrite](#). For simple tasks, see [mod\\_alias](#) and the documentation on [mapping URLs to the filesystem](#).

Finally, before proceeding, be sure to configure [mod\\_rewrite](#)'s log level to one of the trace levels using the [LogLevel](#) directive. Although this can give an overwhelming amount of information, it is indispensable in debugging problems with [mod\\_rewrite](#) configuration, since it will tell you exactly how each rule is processed.



mod\_rewrite uses the [Perl Compatible Regular Expression](#) vocabulary. In this document, we do not attempt to provide a detailed reference to regular expressions. For that, we recommend the [PCRE man pages](#), the [Perl regular expression man page](#), and [Mastering Regular Expressions, by Jeffrey Friedl](#).

In this document, we attempt to provide enough of a regex vocabulary to get you started, without being overwhelming, in the hope that [RewriteRules](#) will be scientific formulae, rather than magical incantations.

## Regex vocabulary

The following are the minimal building blocks you will need, in order to write regular expressions and [RewriteRules](#). They certainly do not represent a complete regular expression vocabulary, but they are a good place to start, and should help you read basic regular expressions, as well as write your own.

Character	Meaning	Example
.	Matches any single character	c.t will match cat, cot, cut, etc.
+	Repeats the previous match one or more times	a+ matches a, aa, aaa, etc
*	Repeats the previous match zero or more times.	a* matches all the same things a+ matches, but will also match an empty string.
?	Makes the match optional.	colou?r will match color and colour.
^	Called an anchor, matches the beginning of the string	^a matches a string that begins with a

\$	The other anchor, this matches the end of the string.	a\$ matches a string that ends with a.
( )	Groups several characters into a single unit, and captures a match for use in a backreference.	(ab)+ matches ababab - that is, the + applies to the group. For more on backreferences see <a href="#">below</a> .
[ ]	A character class - matches one of the characters	c[ua]t matches cut, cot or cat.
[^ ]	Negative character class - matches any character not specified	c[^/]t matches cat or c=t but not c/t

In [mod\\_rewrite](#) the ! character can be used before a regular expression to negate it. This is, a string will be considered to have matched only if it does not match the rest of the expression.

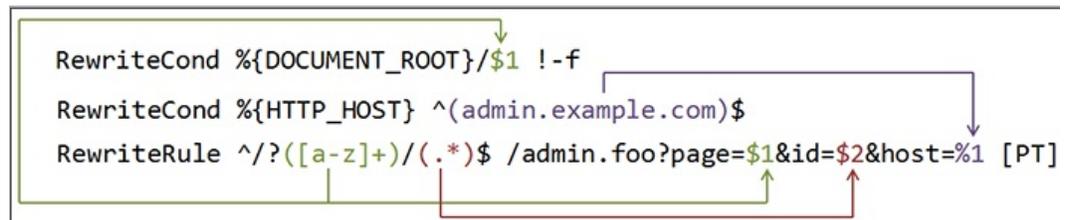
## Regex Back-Reference Availability

One important thing here has to be remembered: Whenever you use parentheses in *Pattern* or in one of the *CondPattern*, back-references are internally created which can be used with the strings \$N and %N (see below). These are available for creating the *Substitution* parameter of a [RewriteRule](#) or the *TestString* parameter of a [RewriteCond](#).

Captures in the [RewriteRule](#) patterns are (counterintuitively) available to all preceding [RewriteCond](#) directives, because the [RewriteRule](#) expression is evaluated before the individual conditions.

Figure 1 shows to which locations the back-references are

transferred for expansion as well as illustrating the flow of the RewriteRule, RewriteCond matching. In the next chapters, we will be exploring how to use these back-references, so do not fret if it seems a bit alien to you at first.



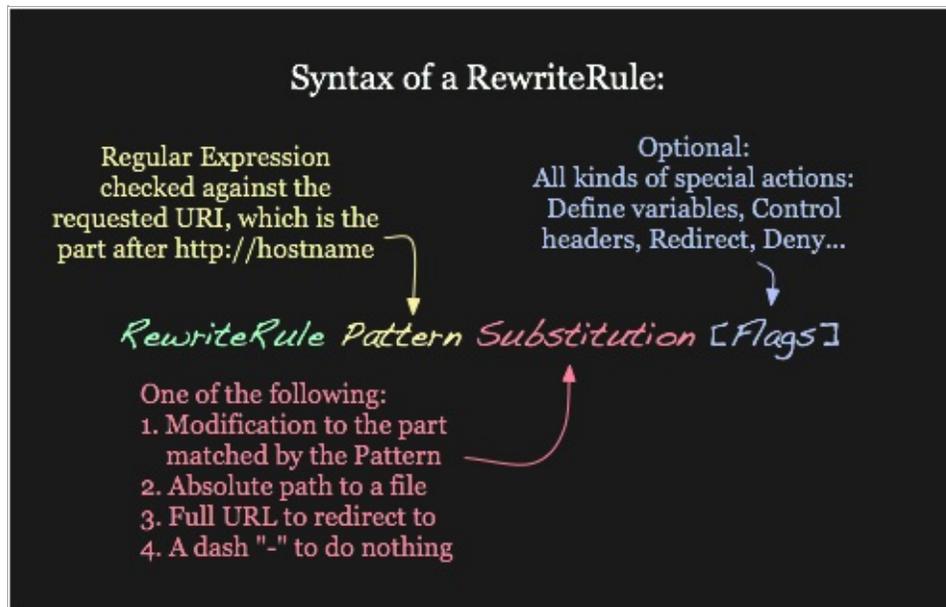
**Figure 1:** The back-reference flow through a rule. In this example, a request for `/test/1234` would be transformed into `/admin.foo?page=test&id=1234&host=admin.example.com`.



A **RewriteRule** consists of three arguments separated by spaces. The arguments are

1. *Pattern*: which incoming URLs should be affected by the rule;
2. *Substitution*: where should the matching requests be sent;
3. *[flags]*: options affecting the rewritten request.

The *Pattern* is a **regular expression**. It is initially (for the first rewrite rule or until a substitution occurs) matched against the URL-path of the incoming request (the part after the hostname but before any question mark indicating the beginning of a query string) or, in per-directory context, against the request's path relative to the directory for which the rule is defined. Once a substitution has occurred, the rules that follow are matched against the substituted value.



**Figure 2:** Syntax of the RewriteRule directive.

The *Substitution* can itself be one of three things:

**A full filesystem path to a resource**

```
RewriteRule "^/games" "/usr/local/games/"
```

This maps a request to an arbitrary location on your filesystem, much like the [Alias](#) directive.

### A web-path to a resource

```
RewriteRule "^/foo$" "/bar"
```

If [DocumentRoot](#) is set to `/usr/local/apache2/htdocs`, then this directive would map requests for `http://example.com/foo` to the path `/usr/local/apache2/htdocs/bar`.

### An absolute URL

```
RewriteRule "^/product/view$" "http://si
```

This tells the client to make a new request for the specified URL.

The *Substitution* can also contain *back-references* to parts of the incoming URL-path matched by the *Pattern*. Consider the following:

```
RewriteRule "^/product/(.*)/view$" "/var/web
```

The variable `$1` will be replaced with whatever text was matched by the expression inside the parenthesis in the *Pattern*. For example, a request for `http://example.com/product/r14df/view` will be mapped

to the path `/var/web/productdb/r14df`.

If there is more than one expression in parenthesis, they are available in order in the variables `$1`, `$2`, `$3`, and so on.



## Rewrite Flags

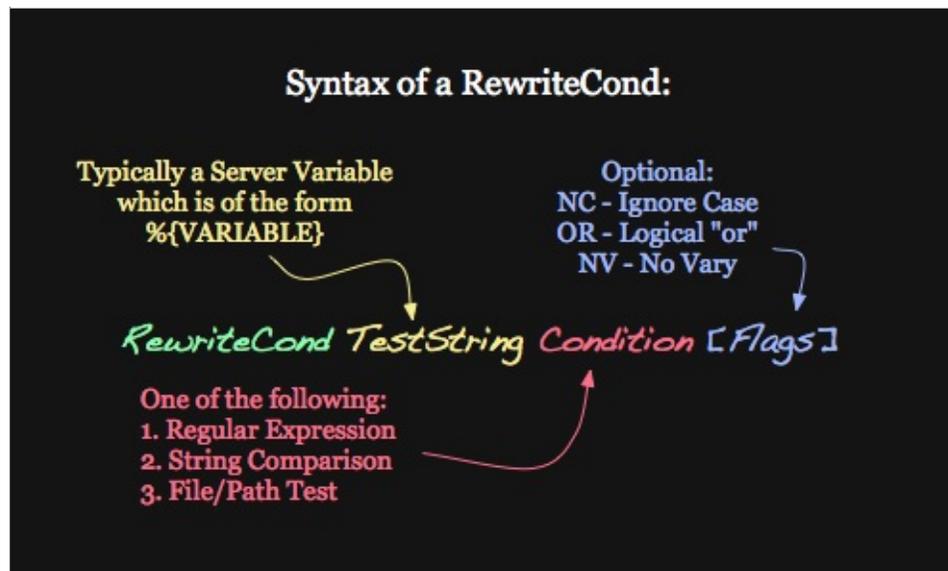
The behavior of a [RewriteRule](#) can be modified by the application of one or more flags to the end of the rule. For example, the matching behavior of a rule can be made case-insensitive by the application of the [NC] flag:

```
RewriteRule "^puppy.html" "smallldog.html" [NC]
```

For more details on the available flags, their meanings, and examples, see the [Rewrite Flags](#) document.



One or more [RewriteCond](#) directives can be used to restrict the types of requests that will be subject to the following [RewriteRule](#). The first argument is a variable describing a characteristic of the request, the second argument is a [regular expression](#) that must match the variable, and a third optional argument is a list of flags that modify how the match is evaluated.



**Figure 3:** Syntax of the RewriteCond directive

For example, to send all requests from a particular IP range to a different server, you could use:

```
RewriteCond "%{REMOTE_ADDR}" "^10\.2\."
RewriteRule "(.*)" "http://intranet.example
```

When more than one [RewriteCond](#) is specified, they must all match for the [RewriteRule](#) to be applied. For example, to deny requests that contain the word "hack" in their query string, unless they also contain a cookie containing the word "go", you could use:

```
RewriteCond "%{QUERY_STRING}" "hack"  
RewriteCond "%{HTTP_COOKIE}" "!go"  
RewriteRule "." "-" [F]
```

Notice that the exclamation mark specifies a negative match, so the rule is only applied if the cookie does not contain "go".

Matches in the regular expressions contained in the [RewriteConds](#) can be used as part of the *Substitution* in the [RewriteRule](#) using the variables %1, %2, etc. For example, this will direct the request to a different directory depending on the hostname used to access the site:

```
RewriteCond "%{HTTP_HOST}" "(.*)" "  
RewriteRule "^/(.*)" "/sites/%1/$1"
```

If the request was for `http://example.com/foo/bar`, then %1 would contain `example.com` and \$1 would contain `foo/bar`.



## RewriteMap

The `RewriteMap` directive provides a way to call an external function, so to speak, to do your rewriting for you. This is discussed in greater detail in the [RewriteMap supplementary documentation](#).



Rewriting is typically configured in the main server configuration setting (outside any `<Directory>` section) or inside `<VirtualHost>` containers. This is the easiest way to do rewriting and is recommended. It is possible, however, to do rewriting inside `<Directory>` sections or `.htaccess files` at the expense of some additional complexity. This technique is called per-directory rewrites.

The main difference with per-server rewrites is that the path prefix of the directory containing the `.htaccess` file is stripped before matching in the `RewriteRule`. In addition, the `RewriteBase` should be used to assure the request is properly mapped.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# RewriteRule Flags

This document discusses the flags which are available to the [RewriteRule](#) directive, providing detailed explanations and examples.



## See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Redirection and remapping](#)
- [Controlling access](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Using RewriteMap](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



A [RewriteRule](#) can have its behavior modified by one or more flags. Flags are included in square brackets at the end of the rule, and multiple flags are separated by commas.

```
RewriteRule pattern target [Flag1,Flag2,Flag3]
```

Each flag (with a few exceptions) has a short form, such as CO, as well as a longer form, such as cookie. While it is most common to use the short form, it is recommended that you familiarize yourself with the long form, so that you remember what each flag is supposed to do. Some flags take one or more arguments. Flags are not case sensitive.

Flags that alter metadata associated with the request (T=, H=, E=) have no affect in per-directory and htaccess context, when a substitution (other than '-') is performed during the same round of rewrite processing.

Presented here are each of the available flags, along with an example of how you might use them.



The [B] flag instructs [RewriteRule](#) to escape non-alphanumeric characters before applying the transformation.

In 2.4.26 and later, you can limit the escaping to specific characters in backreferences by listing them: [B=#? ; ]. Note: The space character can be used in the list of characters to escape, but it cannot be the last character in the list.

mod\_rewrite has to unescape URLs before mapping them, so backreferences are unescaped at the time they are applied. Using the B flag, non-alphanumeric characters in backreferences will be escaped. For example, consider the rule:

```
RewriteRule "^search/(.*)$" "/search.php?term=$1" [B]
```

Given a search term of 'x & y/z', a browser will encode it as 'x%20%26%20y%2Fz', making the request 'search/x%20%26%20y%2Fz'. Without the B flag, this rewrite rule will map to 'search.php?term=x & y/z', which isn't a valid URL, and so would be encoded as search.php?term=x%20&y%2Fz=, which is not what was intended.

With the B flag set on this same rule, the parameters are re-encoded before being passed on to the output URL, resulting in a correct mapping to /search.php?term=x%20%26%20y%2Fz.

```
RewriteRule "^search/(.*)$" "/search.php?term=$1" [B]
```

Note that you may also need to set [AllowEncodedSlashes](#) to On to get this particular example to work, as httpd does not allow encoded slashes in URLs, and returns a 404 if it sees one.

This escaping is particularly necessary in a proxy situation, when the backend may break if presented with an unescaped URL.

An alternative to this flag is using a [RewriteCond](#) to capture against  `%{THE_REQUEST}` which will capture strings in the encoded form.



The [BNP] flag instructs `RewriteRule` to escape the space character in a backreference to %20 rather than '+'. Useful when the backreference will be used in the path component rather than the query string.

This flag is available in version 2.4.26 and later.



The [C] or [chain] flag indicates that the [RewriteRule](#) is chained to the next rule. That is, if the rule matches, then it is processed as usual and control moves on to the next rule. However, if it does not match, then the next rule, and any other rules that are chained together, are skipped.



The [CO], or [cookie] flag, allows you to set a cookie when a particular [RewriteRule](#) matches. The argument consists of three required fields and four optional fields.

The full syntax for the flag, including all attributes, is as follows:

```
[CO=NAME:VALUE:DOMAIN:lifetime:path:secure:httponly]
```

If a literal ':' character is needed in any of the cookie fields, an alternate syntax is available. To opt-in to the alternate syntax, the cookie "Name" should be preceded with a ';' character, and field separators should be specified as ';':

```
[CO=;NAME;VALUE:MOREVALUE;DOMAIN;lifetime;path;secure;httponly]
```

You must declare a name, a value, and a domain for the cookie to be set.

### **Domain**

The domain for which you want the cookie to be valid. This may be a hostname, such as `www.example.com`, or it may be a domain, such as `.example.com`. It must be at least two parts separated by a dot. That is, it may not be merely `.com` or `.net`. Cookies of that kind are forbidden by the cookie security model.

You may optionally also set the following values:

### **Lifetime**

The time for which the cookie will persist, in minutes. A value of 0 indicates that the cookie will persist only for the current browser session. This is the default value if none is specified.

## Path

The path, on the current website, for which the cookie is valid, such as `/customers/` or `/files/download/`.

By default, this is set to `/` - that is, the entire website.

## Secure

If set to `secure`, `true`, or `1`, the cookie will only be permitted to be translated via secure (`https`) connections.

## httponly

If set to `HttpOnly`, `true`, or `1`, the cookie will have the `HttpOnly` flag set, which means that the cookie is inaccessible to JavaScript code on browsers that support this feature.

Consider this example:

```
RewriteEngine On  
RewriteRule "^/index\.html" "-" [CO=frontdoor]
```

In the example given, the rule doesn't rewrite the request. The `"-"` rewrite target tells `mod_rewrite` to pass the request through unchanged. Instead, it sets a cookie called `'frontdoor'` to a value of `'yes'`. The cookie is valid for any host in the `.example.com` domain. It is set to expire in 1440 minutes (24 hours) and is returned for all URIs.



The DPI flag causes the PATH\_INFO portion of the rewritten URI to be discarded.

This flag is available in version 2.2.12 and later.

In per-directory context, the URI each RewriteRule compares against is the concatenation of the current values of the URI and PATH\_INFO.

The current URI can be the initial URI as requested by the client, the result of a previous round of mod\_rewrite processing, or the result of a prior rule in the current round of mod\_rewrite processing.

In contrast, the PATH\_INFO that is appended to the URI before each rule reflects only the value of PATH\_INFO before this round of mod\_rewrite processing. As a consequence, if large portions of the URI are matched and copied into a substitution in multiple RewriteRule directives, without regard for which parts of the URI came from the current PATH\_INFO, the final URI may have multiple copies of PATH\_INFO appended to it.

Use this flag on any substitution where the PATH\_INFO that resulted from the previous mapping of this request to the filesystem is not of interest. This flag permanently forgets the PATH\_INFO established before this round of mod\_rewrite processing began. PATH\_INFO will not be recalculated until the current round of mod\_rewrite processing completes. Subsequent rules during this round of processing will see only the direct result of substitutions, without any PATH\_INFO appended.



With the [E], or [env] flag, you can set the value of an environment variable. Note that some environment variables may be set after the rule is run, thus unsetting what you have set. See [the Environment Variables document](#) for more details on how Environment variables work.

The full syntax for this flag is:

```
[E=VAR:VAL]  
[E=!VAR]
```

VAL may contain backreferences (\$N or %N) which are expanded.

Using the short form

```
[E=VAR]
```

you can set the environment variable named VAR to an empty value.

The form

```
[E=!VAR]
```

allows to unset a previously set environment variable named VAR.

Environment variables can then be used in a variety of contexts, including CGI programs, other RewriteRule directives, or CustomLog directives.

The following example sets an environment variable called 'image' to a value of '1' if the requested URI is an image file. Then, that environment variable is used to exclude those requests from the access log.

```
RewriteRule "\.(png|gif|jpg)$" "-" [E=image  
CustomLog "logs/access_log" combined env=!ir
```

Note that this same effect can be obtained using [SetEnvIf](#). This technique is offered as an example, not as a recommendation.



---

Using the [END] flag terminates not only the current round of rewrite processing (like [L]) but also prevents any subsequent rewrite processing from occurring in per-directory (htaccess) context.

This does not apply to new requests resulting from external redirects.



## Forbidden

Using the [F] flag causes the server to return a 403 Forbidden status code to the client. While the same behavior can be accomplished using the [Deny](#) directive, this allows more flexibility in assigning a Forbidden status.

The following rule will forbid .exe files from being downloaded from your server.

```
RewriteRule "\.exe" "-" [F]
```

This example uses the "-" syntax for the rewrite target, which means that the requested URI is not modified. There's no reason to rewrite to another URI, if you're going to forbid the request.

When using [F], an [L] is implied - that is, the response is returned immediately, and no further rules are evaluated.



The [G] flag forces the server to return a 410 Gone status with the response. This indicates that a resource used to be available, but is no longer available.

As with the [F] flag, you will typically use the "-" syntax for the rewrite target when using the [G] flag:

```
RewriteRule "oldproduct" "-" [G,NC]
```

When using [G], an [L] is implied - that is, the response is returned immediately, and no further rules are evaluated.



Forces the resulting request to be handled with the specified handler. For example, one might use this to force all files without a file extension to be parsed by the php handler:

```
RewriteRule "!\. " "-" [H=application/x-httpd
```

The regular expression above - `!\.` - will match any request that does not contain the literal `.` character.

This can be also used to force the handler based on some conditions. For example, the following snippet used in per-server context allows `.php` files to be *displayed* by `mod_php` if they are requested with the `.phps` extension:

```
RewriteRule "^(/source/.\+\.php)s$" "$1" [H=
```

The regular expression above - `^(/source/.\+\.php)s$` - will match any request that starts with `/source/` followed by 1 or n characters followed by `.phps` literally. The backreference `$1` refers to the captured match within parenthesis of the regular expression.



The [L] flag causes `mod_rewrite` to stop processing the rule set. In most contexts, this means that if the rule matches, no further rules will be processed. This corresponds to the `last` command in Perl, or the `break` command in C. Use this flag to indicate that the current rule should be applied immediately without considering further rules.

If you are using `RewriteRule` in either `.htaccess` files or in `<Directory>` sections, it is important to have some understanding of how the rules are processed. The simplified form of this is that once the rules have been processed, the rewritten request is handed back to the URL parsing engine to do what it may wish with it. It is possible that as the rewritten request is handled, the `.htaccess` file or `<Directory>` section may be encountered again, and thus the ruleset may be run again from the start. Most commonly this will happen if one of the rules causes a redirect - either internal or external - causing the request process to start over.

It is therefore important, if you are using `RewriteRule` directives in one of these contexts, that you take explicit steps to avoid rules looping, and not count solely on the [L] flag to terminate execution of a series of rules, as shown below.

An alternative flag, [END], can be used to terminate not only the current round of rewrite processing but prevent any subsequent rewrite processing from occurring in per-directory (htaccess) context. This does not apply to new requests resulting from external redirects.

The example given here will rewrite any request to `index.php`, giving the original request as a query string argument to `index.php`, however, the `RewriteCond` ensures that if the

request is already for `index.php`, the `RewriteRule` will be skipped.

```
RewriteBase "/"  
RewriteCond "%{REQUEST_URI}" "!=/index.php"  
RewriteRule "^(.*)" "/index.php?req=$1" [L,F
```



The [N] flag causes the ruleset to start over again from the top, using the result of the ruleset so far as a starting point. Use with extreme caution, as it may result in loop.

The [Next] flag could be used, for example, if you wished to replace a certain string or letter repeatedly in a request. The example shown here will replace A with B everywhere in a request, and will continue doing so until there are no more As to be replaced.

```
RewriteRule "(.*)A(.*)" "$1B$2" [N]
```

You can think of this as a `while` loop: While this pattern still matches (i.e., while the URI still contains an A), perform this substitution (i.e., replace the A with a B).

In 2.4.8 and later, this module returns an error after 32,000 iterations to protect against unintended looping. An alternative maximum number of iterations can be specified by adding to the N flag.

```
# Be willing to replace 1 character in each  
RewriteRule "(.+)[><;]$" "$1" [N=64000]  
# ... or, give up if after 10 loops  
RewriteRule "(.+)[><;]$" "$1" [N=10]
```



Use of the [NC] flag causes the `RewriteRule` to be matched in a case-insensitive manner. That is, it doesn't care whether letters appear as upper-case or lower-case in the matched URI.

In the example below, any request for an image file will be proxied to your dedicated image server. The match is case-insensitive, so that `.jpg` and `.JPG` files are both acceptable, for example.

```
RewriteRule "(.*\.(jpg|gif|png))$" "http://:
```



By default, special characters, such as & and ?, for example, will be converted to their hexcode equivalent. Using the [NE] flag prevents that from happening.

```
RewriteRule "^/anchor/(.+)" "/bigpage.html#s
```

The above example will redirect /anchor/xyz to /bigpage.html#xyz. Omitting the [NE] will result in the # being converted to its hexcode equivalent, %23, which will then result in a 404 Not Found error condition.



Use of the [NS] flag prevents the rule from being used on subrequests. For example, a page which is included using an SSI (Server Side Include) is a subrequest, and you may want to avoid rewrites happening on those subrequests. Also, when `mod_dir` tries to find out information about possible directory default files (such as `index.html` files), this is an internal subrequest, and you often want to avoid rewrites on such subrequests. On subrequests, it is not always useful, and can even cause errors, if the complete set of rules are applied. Use this flag to exclude problematic rules.

To decide whether or not to use this rule: if you prefix URLs with CGI-scripts, to force them to be processed by the CGI-script, it's likely that you will run into problems (or significant overhead) on sub-requests. In these cases, use this flag.

Images, javascript files, or css files, loaded as part of an HTML page, are not subrequests - the browser requests them as separate HTTP requests.



Use of the [P] flag causes the request to be handled by [mod\\_proxy](#), and handled via a proxy request. For example, if you wanted all image requests to be handled by a back-end image server, you might do something like the following:

```
RewriteRule "/(.*)\.(jpg|gif|png)$" "http://
```

Use of the [P] flag implies [L] - that is, the request is immediately pushed through the proxy, and any following rules will not be considered.

You must make sure that the substitution string is a valid URI (typically starting with `http://hostname`) which can be handled by the [mod\\_proxy](#). If not, you will get an error from the proxy module. Use this flag to achieve a more powerful implementation of the [ProxyPass](#) directive, to map remote content into the namespace of the local server.

### Security Warning

Take care when constructing the target URL of the rule, considering the security impact from allowing the client influence over the set of URLs to which your server will act as a proxy. Ensure that the scheme and hostname part of the URL is either fixed, or does not allow the client undue influence.

### Performance warning

Using this flag triggers the use of [mod\\_proxy](#), without handling of persistent connections. This means the performance of your proxy will be better if you set it up with [ProxyPass](#) or [ProxyPassMatch](#)

This is because this flag triggers the use of the default worker, which does not handle connection pooling/reuse.

Avoid using this flag and prefer those directives, whenever you can.

Note: [mod\\_proxy](#) must be enabled in order to use this flag.



The target (or substitution string) in a RewriteRule is assumed to be a file path, by default. The use of the [PT] flag causes it to be treated as a URI instead. That is to say, the use of the [PT] flag causes the result of the [RewriteRule](#) to be passed back through URL mapping, so that location-based mappings, such as [Alias](#), [Redirect](#), or [ScriptAlias](#), for example, might have a chance to take effect.

If, for example, you have an [Alias](#) for /icons, and have a [RewriteRule](#) pointing there, you should use the [PT] flag to ensure that the [Alias](#) is evaluated.

```
Alias "/icons" "/usr/local/apache/icons"  
RewriteRule "/pics/(.+)\.jpg$" "/icons/$1.g:
```

Omission of the [PT] flag in this case will cause the Alias to be ignored, resulting in a 'File not found' error being returned.

The PT flag implies the L flag: rewriting will be stopped in order to pass the request to the next phase of processing.

Note that the PT flag is implied in per-directory contexts such as [<Directory>](#) sections or in .htaccess files. The only way to circumvent that is to rewrite to -.



When the replacement URI contains a query string, the default behavior of `RewriteRule` is to discard the existing query string, and replace it with the newly generated one. Using the `[QSA]` flag causes the query strings to be combined.

Consider the following rule:

```
RewriteRule "/pages/(.+)" "/page.php?page=$1
```

With the `[QSA]` flag, a request for `/pages/123?one=two` will be mapped to `/page.php?page=123&one=two`. Without the `[QSA]` flag, that same request will be mapped to `/page.php?page=123` - that is, the existing query string will be discarded.



When the requested URI contains a query string, and the target URI does not, the default behavior of `RewriteRule` is to copy that query string to the target URI. Using the `[QSD]` flag causes the query string to be discarded.

This flag is available in version 2.4.0 and later.

Using `[QSD]` and `[QSA]` together will result in `[QSD]` taking precedence.

If the target URI has a query string, the default behavior will be observed - that is, the original query string will be discarded and replaced with the query string in the `RewriteRule` target URI.



By default, the first (left-most) question mark in the substitution delimits the path from the query string. Using the [QSL] flag instructs `RewriteRule` to instead split the two components using the last (right-most) question mark.

This is useful when mapping to files that have literal question marks in their filename. If no query string is used in the substitution, a question mark can be appended to it in combination with this flag.

This flag is available in version 2.4.19 and later.



## Redirect

Use of the [R] flag causes a HTTP redirect to be issued to the browser. If a fully-qualified URL is specified (that is, including `http://servername/`) then a redirect will be issued to that location. Otherwise, the current protocol, servername, and port number will be used to generate the URL sent with the redirect.

Any valid HTTP response status code may be specified, using the syntax `[R=305]`, with a 302 status code being used by default if none is specified. The status code specified need not necessarily be a redirect (3xx) status code. However, if a status code is outside the redirect range (300-399) then the substitution string is dropped entirely, and rewriting is stopped as if the L were used.

In addition to response status codes, you may also specify redirect status using their symbolic names: `temp` (default), `permanent`, or `seeother`.

You will almost always want to use [R] in conjunction with [L] (that is, use `[R,L]`) because on its own, the [R] flag prepends `http://thishost[:thisport]` to the URI, but then passes this on to the next rule in the ruleset, which can often result in 'Invalid URI in request' warnings.



The [S] flag is used to skip rules that you don't want to run. The syntax of the skip flag is [S=N], where N signifies the number of rules to skip (provided the [RewriteRule](#) matches). This can be thought of as a goto statement in your rewrite ruleset. In the following example, we only want to run the [RewriteRule](#) if the requested URI doesn't correspond with an actual file.

```
# Is the request for a non-existent file?
RewriteCond "%{REQUEST_FILENAME}" "!"-f"
RewriteCond "%{REQUEST_FILENAME}" "!"-d"
# If so, skip these two RewriteRules
RewriteRule ".?" "-" [S=2]

RewriteRule "(.*\.gif)" "images.php?$1"
RewriteRule "(.*\.html)" "docs.php?$1"
```

This technique is useful because a [RewriteCond](#) only applies to the [RewriteRule](#) immediately following it. Thus, if you want to make a RewriteCond apply to several RewriteRules, one possible technique is to negate those conditions and add a RewriteRule with a [Skip] flag. You can use this to make pseudo if-then-else constructs: The last rule of the then-clause becomes skip=N, where N is the number of rules in the else-clause:

```
# Does the file exist?
RewriteCond "%{REQUEST_FILENAME}" "!"-f"
RewriteCond "%{REQUEST_FILENAME}" "!"-d"
# Create an if-then-else construct by skipping:
RewriteRule ".?" "-" [S=3]

# IF the file exists, then:
    RewriteRule "(.*\.gif)" "images.php?$1"
    RewriteRule "(.*\.html)" "docs.php?$1"
# Skip past the "else" stanza.
```

```
    RewriteRule ".?" "-" [S=1]
# ELSE...
    RewriteRule "(.*)" "404.php?file=$1"
# END
```

It is probably easier to accomplish this kind of configuration using the `<If>`, `<ElseIf>`, and `<Else>` directives instead.



Sets the MIME type with which the resulting response will be sent. This has the same effect as the [AddType](#) directive.

For example, you might use the following technique to serve Perl source code as plain text, if requested in a particular way:

```
# Serve .pl files as plain text
RewriteRule "\.pl$" "-" [T=text/plain]
```

Or, perhaps, if you have a camera that produces jpeg images without file extensions, you could force those images to be served with the correct MIME type by virtue of their file names:

```
# Files with 'IMG' in the name are jpeg images
RewriteRule "IMG" "-" [T=image/jpeg]
```

Please note that this is a trivial example, and could be better done using [<FilesMatch>](#) instead. Always consider the alternate solutions to a problem before resorting to rewrite, which will invariably be a less efficient solution than the alternatives.

If used in per-directory context, use only - (dash) as the substitution *for the entire round of mod\_rewrite processing*, otherwise the MIME-type set with this flag is lost due to an internal re-processing (including subsequent rounds of mod\_rewrite processing). The L flag can be useful in this context to end the *current* round of mod\_rewrite processing.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer](#)

# Developing modules for the Apache HTTP Server 2.4

This document explains how you can develop modules for the Apache HTTP Server 2.4



## See also

[Request Processing in Apache 2.4](#)

[Apache 2.x Hook Functions](#)



---

## What we will be discussing in this document

This document will discuss how you can create modules for the Apache HTTP Server 2.4, by exploring an example module called `mod_example`. In the first part of this document, the purpose of this module will be to calculate and print out various digest values for existing files on your web server, whenever we access the URL `http://hostname/filename.sum`. For instance, if we want to know the MD5 digest value of the file located at `http://www.example.com/index.html`, we would visit `http://www.example.com/index.html.sum`.

In the second part of this document, which deals with configuration directive and context awareness, we will be looking at a module that simply writes out its own configuration to the client.

## Prerequisites

First and foremost, you are expected to have a basic knowledge of how the C programming language works. In most cases, we will try to be as pedagogical as possible and link to documents describing the functions used in the examples, but there are also many cases where it is necessary to either just assume that "it works" or do some digging yourself into what the hows and whys of various function calls.

Lastly, you will need to have a basic understanding of how modules are loaded and configured in the Apache HTTP Server, as well as how to get the headers for Apache if you do not have them already, as these are needed for compiling new modules.

## Compiling your module

To compile the source code we are building in this document, we

will be using [APXS](#). Assuming your source file is called `mod_example.c`, compiling, installing and activating the module is as simple as:

```
apxs -i -a -c mod_example.c
```



## Defining a module



Every module starts with the same declaration, or name tag if you will, that defines a module as *a separate entity within Apache*:

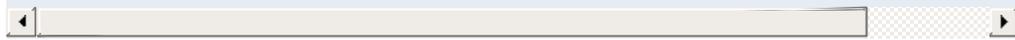
```
module AP_MODULE_DECLARE_DATA example_mod
{
    STANDARD20_MODULE_STUFF,
    create_dir_conf, /* Per-directory configuration
    merge_dir_conf, /* Merge handler for per-directory
    create_svr_conf, /* Per-server configuration
    merge_svr_conf, /* Merge handler for per-server
    directives, /* Any directives we may want
    register_hooks /* Our hook registering
};
```

This bit of code lets the server know that we have now registered a new module in the system, and that its name is `example_module`. The name of the module is used primarily for two things:

- Letting the server know how to load the module using the `LoadModule`
- Setting up a namespace for the module to use in configurations

For now, we're only concerned with the first purpose of the module name, which comes into play when we need to load the module:

```
LoadModule example_module modules/mod_examp
```



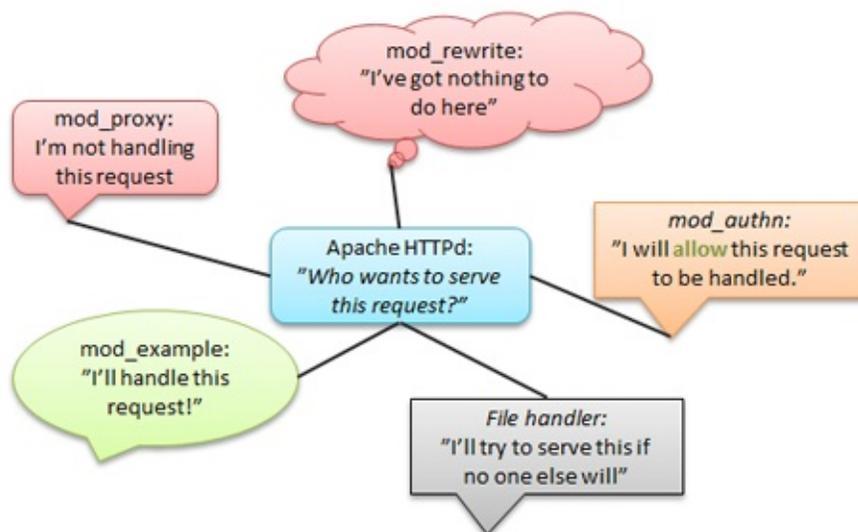
In essence, this tells the server to open up `mod_example.so` and look for a module called `example_module`.

Within this name tag of ours is also a bunch of references to how we would like to handle things: Which directives do we respond to in a configuration file or `.htaccess`, how do we operate within specific contexts, and what handlers are we interested in registering with the Apache HTTP service. We'll return to all these elements later in this document.



## An introduction to hooks

When handling requests in Apache HTTP Server 2.4, the first thing you will need to do is create a hook into the request handling process. A hook is essentially a message telling the server that you are willing to either serve or at least take a glance at certain requests given by clients. All handlers, whether it's `mod_rewrite`, `mod_authn_*`, `mod_proxy` and so on, are hooked into specific parts of the request process. As you are probably aware, modules serve different purposes; Some are authentication/authorization handlers, others are file or script handlers while some third modules rewrite URIs or proxies content. Furthermore, in the end, it is up to the user of the server how and when each module will come into place. Thus, the server itself does not presume to know which module is responsible for handling a specific request, and will ask each module whether they have an interest in a given request or not. It is then up to each module to either gently decline serving a request, accept serving it or flat out deny the request from being served, as authentication/authorization modules do:



To make it a bit easier for handlers such as our `mod_example` to know whether the client is requesting content we should handle or not, the server has directives for hinting to modules whether their

assistance is needed or not. Two of these are [AddHandler](#) and [SetHandler](#). Let's take a look at an example using [AddHandler](#). In our example case, we want every request ending with `.sum` to be served by `mod_example`, so we'll add a configuration directive that tells the server to do just that:

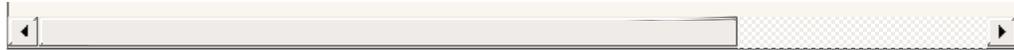
```
AddHandler example-handler .sum
```

What this tells the server is the following: *Whenever we receive a request for a URI ending in `.sum`, we are to let all modules know that we are looking for whoever goes by the name of "example-handler"*. Thus, when a request is being served that ends in `.sum`, the server will let all modules know, that this request should be served by "example-handler". As you will see later, when we start building `mod_example`, we will check for this handler tag relayed by `AddHandler` and reply to the server based on the value of this tag.

## Hooking into httpd

To begin with, we only want to create a simple handler, that replies to the client browser when a specific URL is requested, so we won't bother setting up configuration handlers and directives just yet. Our initial module definition will look like this:

```
module AP_MODULE_DECLARE_DATA example_mod
{
    STANDARD20_MODULE_STUFF,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    register_hooks /* Our hook registering
};
```



This lets the server know that we are not interested in anything fancy, we just want to hook onto the requests and possibly handle some of them.

The reference in our example declaration, `register_hooks` is the name of a function we will create to manage how we hook onto the request process. In this example module, the function has just one purpose; To create a simple hook that gets called after all the rewrites, access control etc has been handled. Thus, we will let the server know, that we want to hook into its process as one of the last modules:

```
static void register_hooks(apr_pool_t *pool
{
    /* Create a hook in the request handler
    ap_hook_handler(example_handler, NULL,
}

```

The `example_handler` reference is the function that will handle the request. We will discuss how to create a handler in the next chapter.

## Other useful hooks

Hooking into the request handling phase is but one of many hooks that you can create. Some other ways of hooking are:

- `ap_hook_child_init`: Place a hook that executes when a child process is spawned (commonly used for initializing modules after the server has forked)
- `ap_hook_pre_config`: Place a hook that executes before any configuration data has been read (very early hook)

- `ap_hook_post_config`: Place a hook that executes after configuration has been parsed, but before the server has forked
- `ap_hook_translate_name`: Place a hook that executes when a URI needs to be translated into a filename on the server (think `mod_rewrite`)
- `ap_hook_quick_handler`: Similar to `ap_hook_handler`, except it is run before any other request hooks (translation, auth, fixups etc)
- `ap_hook_log_transaction`: Place a hook that executes when the server is about to add a log entry of the current request



## Creating a handler

A handler is essentially a function that receives a callback when a request to the server is made. It is passed a record of the current request (how it was made, which headers and requests were passed along, who's giving the request and so on), and is put in charge of either telling the server that it's not interested in the request or handle the request with the tools provided.

### A simple "Hello, world!" handler

Let's start off by making a very simple request handler that does the following:

1. Check that this is a request that should be served by "example-handler"
2. Set the content type of our output to `text/html`
3. Write "Hello, world!" back to the client browser
4. Let the server know that we took care of this request and everything went fine

In C code, our example handler will now look like this:

```
static int example_handler(request_rec *r)
{
    /* First off, we need to check if this
     * If it is, we accept it and do our th
     * and the server will try somewhere el
     */
    if (!r->handler || strcmp(r->handler, "

    /* Now that we are handling this request
     * To do so, we must first set the appr
     */
    ap_set_content_type(r, "text/html");
    ap_rprintf(r, "Hello, world!");
}
```

```
    /* Lastly, we must tell the server that
     * We do so by simply returning the val
     */
    return OK;
}
```

Now, we put all we have learned together and end up with a program that looks like [mod\\_example\\_1.c](#) . The functions used in this example will be explained later in the section ["Some useful functions you should know"](#).

## The request\_rec structure

The most essential part of any request is the *request record* . In a call to a handler function, this is represented by the `request_rec*` structure passed along with every call that is made. This struct, typically just referred to as `r` in modules, contains all the information you need for your module to fully process any HTTP request and respond accordingly.

Some key elements of the `request_rec` structure are:

- `r->handler (char*)`: Contains the name of the handler the server is currently asking to do the handling of this request
- `r->method (char*)`: Contains the HTTP method being used, f.x. GET or POST
- `r->filename (char*)`: Contains the translated filename the client is requesting
- `r->args (char*)`: Contains the query string of the request, if any
- `r->headers_in (apr_table_t*)`: Contains all the headers sent by the client
- `r->connection (conn_rec*)`: A record containing

information about the current connection

- `r->user (char*)`: If the URI requires authentication, this is set to the username provided
- `r->useragent_ip (char*)`: The IP address of the client connecting to us
- `r->pool (apr_pool_t*)`: The memory pool of this request. We'll discuss this in the "[Memory management](#)" chapter.

A complete list of all the values contained within the `request_rec` structure can be found in the [httpd.h](#) header file or at

[http://ci.apache.org/projects/httpd/trunk/doxygen/structrequest\\_rec](http://ci.apache.org/projects/httpd/trunk/doxygen/structrequest_rec)

Let's try out some of these variables in another example handler:

```
static int example_handler(request_rec *r)
{
    /* Set the appropriate content type */
    ap_set_content_type(r, "text/html");

    /* Print out the IP address of the client */
    ap_rprintf(r, "<h2>Hello, %s!</h2>", r->useragent_ip);

    /* If we were reached through a GET or POST */
    if ( !strcmp(r->method, "POST") || !strcmp(r->method, "GET") )
        ap_rputs("You used a GET or a POST request.");
    else {
        ap_rputs("You did not use POST or GET.");
    }

    /* Lastly, if there was a query string, print it out */
    if (r->args) {
        ap_rprintf(r, "Your query string was: %s", r->args);
    }
}
```

```
    return OK;
}
```

## Return values

Apache relies on return values from handlers to signify whether a request was handled or not, and if so, whether the request went well or not. If a module is not interested in handling a specific request, it should always return the value `DECLINED`. If it is handling a request, it should either return the generic value `OK`, or a specific HTTP status code, for example:

```
static int example_handler(request_rec *r)
{
    /* Return 404: Not found */
    return HTTP_NOT_FOUND;
}
```

Returning `OK` or a HTTP status code does not necessarily mean that the request will end. The server may still have other handlers that are interested in this request, for instance the logging modules which, upon a successful request, will write down a summary of what was requested and how it went. To do a full stop and prevent any further processing after your module is done, you can return the value `DONE` to let the server know that it should cease all activity on this request and carry on with the next, without informing other handlers.

### General response codes:

- `DECLINED`: We are not handling this request
- `OK`: We handled this request and it went well
- `DONE`: We handled this request and the server should just close this thread without further processing

## HTTP specific return codes (excerpt):

- HTTP\_OK (200): Request was okay
- HTTP\_MOVED\_PERMANENTLY (301): The resource has moved to a new URL
- HTTP\_UNAUTHORIZED (401): Client is not authorized to visit this page
- HTTP\_FORBIDDEN (403): Permission denied
- HTTP\_NOT\_FOUND (404): File not found
- HTTP\_INTERNAL\_SERVER\_ERROR (500): Internal server error (self explanatory)

## Some useful functions you should know

- `ap_rputs(const char *string, request_rec *r)`: Sends a string of text to the client. This is a shorthand version of [ap\\_rwrite](#).

```
ap_rputs("Hello, world!", r);
```

- [ap\\_rprintf](#): This function works just like `printf`, except it sends the result to the client.

```
ap_rprintf(r, "Hello, %s!", r->useragent
```

- [ap\\_set\\_content\\_type](#)(`request_rec *r, const char *type`): Sets the content type of the output you are sending.

```
ap_set_content_type(r, "text/plain"); /*
```

## Memory management

Managing your resources in Apache HTTP Server 2.4 is quite easy, thanks to the memory pool system. In essence, each server, connection and request have their own memory pool that gets cleaned up when its scope ends, e.g. when a request is done or when a server process shuts down. All your module needs to do is latch onto this memory pool, and you won't have to worry about having to clean up after yourself - pretty neat, huh?

In our module, we will primarily be allocating memory for each request, so it's appropriate to use the `r->pool` reference when creating new objects. A few of the functions for allocating memory within a pool are:

- `void* apr_palloc( apr_pool_t *p, apr_size_t size)`: Allocates `size` number of bytes in the pool for you
- `void* apr_pcalloc( apr_pool_t *p, apr_size_t size)`: Allocates `size` number of bytes in the pool for you and sets all bytes to 0
- `char* apr_pstrdup( apr_pool_t *p, const char *s)`: Creates a duplicate of the string `s`. This is useful for copying constant values so you can edit them
- `char* apr_psprintf( apr_pool_t *p, const char *fmt, ...)`: Similar to `sprintf`, except the server supplies you with an appropriately allocated target variable

Let's put these functions into an example handler:

```
static int example_handler(request_rec *r)
{
    const char *original = "You can't edit
    char *copy;
    int *integers;
```

```
/* Allocate space for 10 integer values
integers = apr_pccalloc(r->pool, sizeof(

/* Create a copy of the 'original' vari
copy = apr_pstrdup(r->pool, original);
return OK;
}
```

This is all well and good for our module, which won't need any pre-initialized variables or structures. However, if we wanted to initialize something early on, before the requests come rolling in, we could simply add a call to a function in our `register_hooks` function to sort it out:

```
static void register_hooks(apr_pool_t *pool
{
    /* Call a function that initializes some
example_init_function(pool);
    /* Create a hook in the request handler
ap_hook_handler(example_handler, NULL,
}
```

In this pre-request initialization function we would not be using the same pool as we did when allocating resources for request-based functions. Instead, we would use the pool given to us by the server for allocating memory on a per-process based level.

## Parsing request data

In our example module, we would like to add a feature, that checks which type of digest, MD5 or SHA1 the client would like to see. This could be solved by adding a query string to the request. A query string is typically comprised of several keys and values put together in a string, for instance

valueA=yes&valueB=no&valueC=maybe. It is up to the module itself to parse these and get the data it requires. In our example, we'll be looking for a key called `digest`, and if set to `md5`, we'll produce an MD5 digest, otherwise we'll produce a SHA1 digest.

Since the introduction of Apache HTTP Server 2.4, parsing request data from GET and POST requests have never been easier. All we require to parse both GET and POST data is four simple lines:

```
apr_table_t *GET;  
apr_array_header_t *POST;  
  
ap_args_to_table(r, &GET);  
ap_parse_form_data(r, NULL, &POST, -1, 8192
```

In our specific example module, we're looking for the `digest` value from the query string, which now resides inside a table called `GET`. To extract this value, we need only perform a simple operation:

```
/* Get the "digest" key from the query string  
const char *digestType = apr_table_get(GET,  
  
/* If no key was returned, we will set a default  
if (!digestType) digestType = "sha1";
```

The structures used for the POST and GET data are not exactly the same, so if we were to fetch a value from POST data instead

of the query string, we would have to resort to a few more lines, as outlined in [this example](#) in the last chapter of this document.

## Making an advanced handler

Now that we have learned how to parse form data and manage our resources, we can move on to creating an advanced version of our module, that spits out the MD5 or SHA1 digest of files:

```
static int example_handler(request_rec *r)
{
    int rc, exists;
    apr_finfo_t finfo;
    apr_file_t *file;
    char *filename;
    char buffer[256];
    apr_size_t readBytes;
    int n;
    apr_table_t *GET;
    apr_array_header_t *POST;
    const char *digestType;

    /* Check that the "example-handler" handler
    if (!r->handler || strcmp(r->handler, "example-handler"))
        return HTTP_NOT_FOUND;

    /* Figure out which file is being requested
    filename = apr_pstrdup(r->pool, r->filename);
    filename[strlen(filename)-4] = 0; /* Cut off ".php"

    /* Figure out if the file we request actually exists
    rc = apr_stat(&finfo, filename, APR_FINFO_MIN);
    if (rc == APR_SUCCESS) {
        exists =
        (
            (finfo.filetype != APR_NOFILE)
            && !(finfo.filetype & APR_DIR)
        );
    }
}
```

```

        );
        if (!exists) return HTTP_NOT_FOUND;
    }
    /* If apr_stat failed, we're probably not found
    else return HTTP_FORBIDDEN;

    /* Parse the GET and, optionally, the POST
    ap_args_to_table(r, &GET);
    ap_parse_form_data(r, NULL, &POST, -1, &args);

    /* Set the appropriate content type */
    ap_set_content_type(r, "text/html");

    /* Print a title and some general information
    ap_rprintf(r, "<h2>Information on %s:</h2>");
    ap_rprintf(r, "<b>Size:</b> %u bytes<br>");

    /* Get the digest type the client wants
    digestType = apr_table_get(GET, "digest-type");
    if (!digestType) digestType = "MD5";

    rc = apr_file_open(&file, filename, APR_RDONLY, APR_0755, &pool);
    if (rc == APR_SUCCESS) {

        /* Are we trying to calculate the MD5
        if (!strcasecmp(digestType, "md5"))
            /* Calculate the MD5 sum of the
            union {
                char        chr[16];
                uint32_t    num[4];
            } digest;
            apr_md5_ctx_t md5;
            apr_md5_init(&md5);
            readBytes = 256;
            while ( apr_file_read(file, buf, &readBytes) == APR_SUCCESS)

```

```

        apr_md5_update(&md5, buffer
    }
    apr_md5_final(digest.chr, &md5)

    /* Print out the MD5 digest */
    ap_rputs("<b>MD5: </b><code>",
    for (n = 0; n < APR_MD5_DIGESTS
        ap_rprintf(r, "%08x", diges
    }
    ap_rputs("</code>", r);
    /* Print a link to the SHA1 vers
    ap_rputs("<br/><a href='?digest:
}
else {
    /* Calculate the SHA1 sum of the
    union {
        char        chr[20];
        uint32_t    num[5];
    } digest;
    apr_sha1_ctx_t sha1;
    apr_sha1_init(&sha1);
    readBytes = 256;
    while ( apr_file_read(file, buff
        apr_sha1_update(&sha1, buff
    }
    apr_sha1_final(digest.chr, &sha

    /* Print out the SHA1 digest */
    ap_rputs("<b>SHA1: </b><code>",
    for (n = 0; n < APR_SHA1_DIGESTS
        ap_rprintf(r, "%08x", diges
    }
    ap_rputs("</code>", r);

    /* Print a link to the MD5 vers
    ap_rputs("<br/><a href='?digest:
}

```

```
        apr_file_close(file);  
  
    }  
    /* Let the server know that we responded  
    return OK;  
}
```

This version in its entirety can be found here: [mod\\_example\\_2.c](#).



## Making configuration options

In this next segment of this document, we will turn our eyes away from the digest module and create a new example module, whose only function is to write out its own configuration. The purpose of this is to examine how the server works with configuration, and what happens when you start writing advanced configurations for your modules.

### An introduction to configuration directives

If you are reading this, then you probably already know what a configuration directive is. Simply put, a directive is a way of telling an individual module (or a set of modules) how to behave, such as these directives control how `mod_rewrite` works:

```
RewriteEngine On
RewriteCond "%{REQUEST_URI}" "^/foo/bar"
RewriteRule "^/foo/bar/(.*)$" "/foobar?page=
```

Each of these configuration directives are handled by a separate function, that parses the parameters given and sets up a configuration accordingly.

### Making an example configuration

To begin with, we'll create a basic configuration in C-space:

```
typedef struct {
    int         enabled;           /* Enable or
    const char *path;             /* Some path
    int         typeOfAction;     /* 1 means ac
} example_config;
```

Now, let's put this into perspective by creating a very small module

that just prints out a hard-coded configuration. You'll notice that we use the `register_hooks` function for initializing the configuration values to their defaults:

```
typedef struct {
    int          enabled;          /* Enable or (
    const char *path;             /* Some path
    int          typeOfAction;    /* 1 means ac
} example_config;

static example_config config;

static int example_handler(request_rec *r)
{
    if (!r->handler || strcmp(r->handler, "
ap_set_content_type(r, "text/plain");
ap_rprintf(r, "Enabled: %u\n", config.e
ap_rprintf(r, "Path: %s\n", config.path
ap_rprintf(r, "TypeOfAction: %x\n", con
return OK;
}

static void register_hooks(apr_pool_t *pool
{
    config.enabled = 1;
    config.path = "/foo/bar";
    config.typeOfAction = 0x00;
    ap_hook_handler(example_handler, NULL,
}

/* Define our module as an entity and assign

module AP_MODULE_DECLARE_DATA example_mod
{
    STANDARD20_MODULE_STUFF,
    NULL, /* Per-directory confi
```

```

    NULL,          /* Merge handler for p
    NULL,          /* Per-server configur
    NULL,          /* Merge handler for p
    NULL,          /* Any directives we m
    register_hooks /* Our hook registering
};

```

So far so good. To access our new handler, we could add the following to our configuration:

```

<Location "/example">
    SetHandler example-handler
</Location>

```

When we visit, we'll see our current configuration being spit out by our module.

## Registering directives with the server

What if we want to change our configuration, not by hard-coding new values into the module, but by using either the `httpd.conf` file or possibly a `.htaccess` file? It's time to let the server know that we want this to be possible. To do so, we must first change our *name tag* to include a reference to the configuration directives we want to register with the server:

```

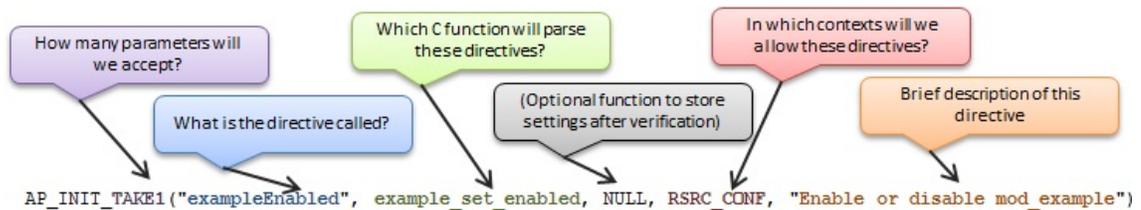
module AP_MODULE_DECLARE_DATA example_mod
{
    STANDARD20_MODULE_STUFF,
    NULL,          /* Per-directory co
    NULL,          /* Merge handler fo
    NULL,          /* Per-server confi
    NULL,          /* Merge handler fo
    example_directives, /* Any directives w
    register_hooks /* Our hook registe

```

```
};
```

This will tell the server that we are now accepting directives from the configuration files, and that the structure called `example_directives` holds information on what our directives are and how they work. Since we have three different variables in our module configuration, we will add a structure with three directives and a NULL at the end:

```
static const command_rec      example_dir
{
    AP_INIT_TAKE1("exampleEnabled", example
    AP_INIT_TAKE1("examplePath", example_se
    AP_INIT_TAKE2("exampleAction", example_
    { NULL }
};
```



As you can see, each directive needs at least 5 parameters set:

1. **AP\_INIT\_TAKE1**: This is a macro that tells the server that this directive takes one and only one argument. If we required two arguments, we could use the macro **AP\_INIT\_TAKE2** and so on (refer to `httpd_conf.h` for more macros).
2. `exampleEnabled`: This is the name of our directive. More precisely, it is what the user must put in his/her configuration in order to invoke a configuration change in our module.
3. `example_set_enabled`: This is a reference to a C function that parses the directive and sets the configuration

accordingly. We will discuss how to make this in the following paragraph.

4. `RSRC_CONF`: This tells the server where the directive is permitted. We'll go into details on this value in the later chapters, but for now, `RSRC_CONF` means that the server will only accept these directives in a server context.
5. `"Enable or disable . . ."`: This is simply a brief description of what the directive does.

*(The "missing" parameter in our definition, which is usually set to `NULL`, is an optional function that can be run after the initial function to parse the arguments have been run. This is usually omitted, as the function for verifying arguments might as well be used to set them.)*

## The directive handler function

Now that we have told the server to expect some directives for our module, it's time to make a few functions for handling these. What the server reads in the configuration file(s) is text, and so naturally, what it passes along to our directive handler is one or more strings, that we ourselves need to recognize and act upon. You'll notice, that since we set our `exampleAction` directive to accept two arguments, its C function also has an additional parameter defined:

```
/* Handler for the "exampleEnabled" directive
const char *example_set_enabled(cmd_parms *cmd, void *p,
{
    if(!strcasecmp(arg, "on")) config.enabled = 1;
    else config.enabled = 0;
    return NULL;
}
```

```

/* Handler for the "examplePath" directive
const char *example_set_path(cmd_parms *cmd
{
    config.path = arg;
    return NULL;
}

/* Handler for the "exampleAction" directive
/* Let's pretend this one takes one argument
/* and we store it in a bit-wise manner. */
const char *example_set_action(cmd_parms *c
{
    if(!strcasecmp(arg1, "file")) config.ty
    else config.typeOfAction = 0x02;

    if(!strcasecmp(arg2, "deny")) config.ty
    else config.typeOfAction += 0x20;
    return NULL;
}

```

## Putting it all together

Now that we have our directives set up, and handlers configured for them, we can assemble our module into one big file:

```

/* mod_example_config_simple.c: */
#include <stdio.h>
#include "apr_hash.h"
#include "ap_config.h"
#include "ap_provider.h"
#include "httpd.h"
#include "http_core.h"
#include "http_config.h"
#include "http_log.h"
#include "http_protocol.h"
#include "http_request.h"

```

```

/*
=====
Our configuration prototype and declaratio
=====
*/
typedef struct {
    int         enabled;        /* Enable or (
    const char *path;          /* Some path
    int         typeOfAction; /* 1 means ac
} example_config;

static example_config config;

/*
=====
Our directive handlers:
=====
*/
/* Handler for the "exampleEnabled" directi
const char *example_set_enabled(cmd_parms *c
{
    if(!strcasecmp(arg, "on")) config.enabled
    else config.enabled = 0;
    return NULL;
}

/* Handler for the "examplePath" directive
const char *example_set_path(cmd_parms *cmd
{
    config.path = arg;
    return NULL;
}

/* Handler for the "exampleAction" directiv
/* Let's pretend this one takes one argumen
/* and we store it in a bit-wise manner. */

```

```

const char *example_set_action(cmd_parms *c
{
    if(!strcasecmp(arg1, "file")) config.ty
    else config.typeOfAction = 0x02;

    if(!strcasecmp(arg2, "deny")) config.ty
    else config.typeOfAction += 0x20;
    return NULL;
}

/*
=====
The directive structure for our name tag:
=====
*/
static const command_rec      example_dir
{
    AP_INIT_TAKE1("exampleEnabled", example
    AP_INIT_TAKE1("examplePath", example_se
    AP_INIT_TAKE2("exampleAction", example_
    { NULL }
};
/*
=====
Our module handler:
=====
*/
static int example_handler(request_rec *r)
{
    if(!r->handler || strcmp(r->handler, "e
    ap_set_content_type(r, "text/plain");
    ap_rprintf(r, "Enabled: %u\n", config.e
    ap_rprintf(r, "Path: %s\n", config.path
    ap_rprintf(r, "TypeOfAction: %x\n", con
    return OK;
}

```

```

/*
=====
The hook registration function (also initi
=====
*/
static void register_hooks(apr_pool_t *pool
{
    config.enabled = 1;
    config.path = "/foo/bar";
    config.typeOfAction = 3;
    ap_hook_handler(example_handler, NULL,
}
/*
=====
Our module name tag:
=====
*/
module AP_MODULE_DECLARE_DATA    example_mod
{
    STANDARD20_MODULE_STUFF,
    NULL,                          /* Per-directory co
    NULL,                          /* Merge handler fo
    NULL,                          /* Per-server confi
    NULL,                          /* Merge handler fo
    example_directives, /* Any directives w
    register_hooks          /* Our hook registe
};

```

In our httpd.conf file, we can now change the hard-coded configuration by adding a few lines:

```

ExampleEnabled On
ExamplePath "/usr/bin/foo"
ExampleAction file allow

```

And thus we apply the configuration, visit `/example` on our web site, and we see the configuration has adapted to what we wrote in our configuration file.



## Introduction to context aware configurations

In Apache HTTP Server 2.4, different URLs, virtual hosts, directories etc can have very different meanings to the user of the server, and thus different contexts within which modules must operate. For example, let's assume you have this configuration set up for `mod_rewrite`:

```
<Directory "/var/www">
    RewriteCond "%{HTTP_HOST}" "^example.com"
    RewriteRule "(.*)" "http://www.example.com"
</Directory>
<Directory "/var/www/sub">
    RewriteRule "^foobar$" "index.php?foobar"
</Directory>
```

In this example, you will have set up two different contexts for `mod_rewrite`:

1. Inside `/var/www`, all requests for `http://example.com` must go to `http://www.example.com`
2. Inside `/var/www/sub`, all requests for `foobar` must go to `index.php?foobar=true`

If `mod_rewrite` (or the entire server for that matter) wasn't context aware, then these rewrite rules would just apply to every and any request made, regardless of where and how they were made, but since the module can pull the context specific configuration straight from the server, it does not need to know itself, which of the directives are valid in this context, since the server takes care of this.

So how does a module get the specific configuration for the server,

directory or location in question? It does so by making one simple call:

```
example_config *config = (example_config*)
```

That's it! Of course, a whole lot goes on behind the scenes, which we will discuss in this chapter, starting with how the server came to know what our configuration looks like, and how it came to be set up as it is in the specific context.

## Our basic configuration setup

In this chapter, we will be working with a slightly modified version of our previous context structure. We will set a context variable that we can use to track which context configuration is being used by the server in various places:

```
typedef struct {
    char    context[256];
    char    path[256];
    int     typeOfAction;
    int     enabled;
} example_config;
```

Our handler for requests will also be modified, yet still very simple:

```
static int example_handler(request_rec *r)
{
    if(!r->handler || strcmp(r->handler, "e
    example_config *config = (example_conf
    ap_set_content_type(r, "text/plain");
    ap_rprintf("Enabled: %u\n", config->ena
    ap_rprintf("Path: %s\n", config->path);
    ap_rprintf("TypeOfAction: %x\n", config
```

```
ap_rprintf("Context: %s\n", config->con
return OK;
}
```

## Choosing a context

Before we can start making our module context aware, we must first define, which contexts we will accept. As we saw in the previous chapter, defining a directive required five elements be set:

```
AP_INIT_TAKE1("exampleEnabled", example_set
```

The `RSRC_CONF` definition told the server that we would only allow this directive in a global server context, but since we are now trying out a context aware version of our module, we should set this to something more lenient, namely the value `ACCESS_CONF`, which lets us use the directive inside `<Directory>` and `<Location>` blocks. For more control over the placement of your directives, you can combine the following restrictions together to form a specific rule:

- `RSRC_CONF`: Allow in `.conf` files (not `.htaccess`) outside `<Directory>` or `<Location>`
- `ACCESS_CONF`: Allow in `.conf` files (not `.htaccess`) inside `<Directory>` or `<Location>`
- `OR_OPTIONS`: Allow in `.conf` files and `.htaccess` when `AllowOverride Options` is set
- `OR_FILEINFO`: Allow in `.conf` files and `.htaccess` when `AllowOverride FileInfo` is set
- `OR_AUTHCFG`: Allow in `.conf` files and `.htaccess` when `AllowOverride AuthConfig` is set

- OR\_INDEXES: Allow in .conf files and .htaccess when AllowOverride Indexes is set
- OR\_ALL: Allow anywhere in .conf files and .htaccess

## Using the server to allocate configuration slots

A much smarter way to manage your configurations is by letting the server help you create them. To do so, we must first start off by changing our *name tag* to let the server know, that it should assist us in creating and managing our configurations. Since we have chosen the per-directory (or per-location) context for our module configurations, we'll add a per-directory creator and merger function reference in our tag:

```
module AP_MODULE_DECLARE_DATA example_mod
{
    STANDARD20_MODULE_STUFF,
    create_dir_conf, /* Per-directory confi
    merge_dir_conf, /* Merge handler for p
    NULL,           /* Per-server configur
    NULL,           /* Merge handler for p
    directives,    /* Any directives we m
    register_hooks /* Our hook registerin
};
```

## Creating new context configurations

Now that we have told the server to help us create and manage configurations, our first step is to make a function for creating new, blank configurations. We do so by creating the function we just referenced in our name tag as the Per-directory configuration handler:

```
void *create_dir_conf(apr_pool_t *pool, cha
```

```
context = context ? context : "(undefined)";
example_config *cfg = apr_pccalloc(pool,
if(cfg) {
    /* Set some default values */
    strcpy(cfg->context, context);
    cfg->enabled = 0;
    cfg->path = "/foo/bar";
    cfg->typeOfAction = 0x11;
}
return cfg;
}
```

## Merging configurations

Our next step in creating a context aware configuration is merging configurations. This part of the process particularly applies to scenarios where you have a parent configuration and a child, such as the following:

```
<Directory "/var/www">
  ExampleEnabled On
  ExamplePath "/foo/bar"
  ExampleAction file allow
</Directory>
<Directory "/var/www/subdir">
  ExampleAction file deny
</Directory>
```

In this example, it is natural to assume that the directory `/var/www/subdir` should inherit the values set for the `/var/www` directory, as we did not specify an `ExampleEnabled` nor an `ExamplePath` for this directory. The server does not presume to know if this is true, but cleverly does the following:

1. Creates a new configuration for `/var/www`

2. Sets the configuration values according to the directives given for `/var/www`
3. Creates a new configuration for `/var/www/subdir`
4. Sets the configuration values according to the directives given for `/var/www/subdir`
5. **Proposes a merge** of the two configurations into a new configuration for `/var/www/subdir`

This proposal is handled by the `merge_dir_conf` function we referenced in our name tag. The purpose of this function is to assess the two configurations and decide how they are to be merged:

```
void *merge_dir_conf(apr_pool_t *pool, void
    example_config *base = (example_config
    example_config *add = (example_config *
    example_config *conf = (example_config

    /* Merge configurations */
    conf->enabled = ( add->enabled == 0 ) ?
    conf->typeOfAction = add->typeOfAction
    strcpy(conf->path, strlen(add->path) ?

    return conf ;
}
```

## Trying out our new context aware configurations

Now, let's try putting it all together to create a new module that is context aware. First off, we'll create a configuration that lets us test how the module works:

```
<Location "/a">
```

```
    SetHandler example-handler
    ExampleEnabled on
    ExamplePath "/foo/bar"
    ExampleAction file allow
</Location>

<Location "/a/b">
    ExampleAction file deny
    ExampleEnabled off
</Location>

<Location "/a/b/c">
    ExampleAction db deny
    ExamplePath "/foo/bar/baz"
    ExampleEnabled on
</Location>
```

Then we'll assemble our module code. Note, that since we are now using our name tag as reference when fetching configurations in our handler, I have added some prototypes to keep the compiler happy:

```
/*$6
+++++
* mod_example_config.c
+++++
*/

#include <stdio.h>
#include "apr_hash.h"
#include "ap_config.h"
#include "ap_provider.h"
#include "httpd.h"
#include "http_core.h"
#include "http_config.h"
```

```

#include "http_log.h"
#include "http_protocol.h"
#include "http_request.h"

/*$1
-----
      Configuration structure
-----
*/

typedef struct
{
    char    context[256];
    char    path[256];
    int     typeOfAction;
    int     enabled;
} example_config;

/*$1
-----
      Prototypes
-----
*/

static int     example_handler(request_rec *
const char    *example_set_enabled(cmd_parms
const char    *example_set_path(cmd_parms *
const char    *example_set_action(cmd_parms
void          *create_dir_conf(apr_pool_t *
void          *merge_dir_conf(apr_pool_t *p
static void    register_hooks(apr_pool_t *po

/*$1
-----
      Configuration directives
-----
*/

```

```

static const command_rec    directives[] =
{
    AP_INIT_TAKE1("exampleEnabled", example
    AP_INIT_TAKE1("examplePath", example_se
    AP_INIT_TAKE2("exampleAction", example_
    { NULL }
};

/*$1
-----
    Our name tag
-----
*/

module AP_MODULE_DECLARE_DATA    example_mo
{
    STANDARD20_MODULE_STUFF,
    create_dir_conf,        /* Per-directory co
    merge_dir_conf,        /* Merge handler fo
    NULL,                    /* Per-server confi
    NULL,                    /* Merge handler fo
    directives,            /* Any directives w
    register_hooks         /* Our hook registe
};

/*
=====
    Hook registration function
=====
*/
static void register_hooks(apr_pool_t *pool
{
    ap_hook_handler(example_handler, NULL,
}

/*

```

```

=====
    Our example web service handler
=====
    */
static int example_handler(request_rec *r)
{
    if(!r->handler || strcmp(r->handler, "e

    /*~~~~~
example_config    *config = (example_co
/*~~~~~

    ap_set_content_type(r, "text/plain");
    ap_rprintf(r, "Enabled: %u\n", config->
    ap_rprintf(r, "Path: %s\n", config->pat
    ap_rprintf(r, "TypeOfAction: %x\n", con
    ap_rprintf(r, "Context: %s\n", config->
    return OK;
}

/*
=====
    Handler for the "exampleEnabled" direct
=====
    */
const char *example_set_enabled(cmd_parms *
{
    /*~~~~~
example_config    *conf = (example_conf
/*~~~~~

    if(conf)
    {
        if(!strcasecmp(arg, "on"))
            conf->enabled = 1;
        else
            conf->enabled = 0;
    }
}

```

```

    }

    return NULL;
}

/*
=====
    Handler for the "examplePath" directive
=====
*/
const char *example_set_path(cmd_parms *cmd
{
    /*~~~~~
    example_config      *conf = (example_conf
    /*~~~~~

    if(conf)
    {
        strcpy(conf->path, arg);
    }

    return NULL;
}

/*
=====
    Handler for the "exampleAction" directive
    Let's pretend this one takes one argument
    and we store it in a bit-wise manner.
=====
*/
const char *example_set_action(cmd_parms *c
{
    /*~~~~~
    example_config      *conf = (example_conf
    /*~~~~~

```

```

if(conf)
{
    {
        if(!strcasecmp(arg1, "file"))
            conf->typeOfAction = 0x01;
        else
            conf->typeOfAction = 0x02;
        if(!strcasecmp(arg2, "deny"))
            conf->typeOfAction += 0x10;
        else
            conf->typeOfAction += 0x20;
    }
}

return NULL;
}

/*
=====
Function for creating new configuration
=====
*/
void *create_dir_conf(apr_pool_t *pool, cha
{
    context = context ? context : "Newly cr

    /*~~~~~
example_config    *cfg = apr_pcalloc(po
    /*~~~~~

if(cfg)
{
    {
        /* Set some default values */
        strcpy(cfg->context, context);
        cfg->enabled = 0;
        memset(cfg->path, 0, 256);
    }
}

```

```
        cfg->typeOfAction = 0x00;
    }
}

return cfg;
}

/*
=====
Merging function for configurations
=====
*/
void *merge_dir_conf(apr_pool_t *pool, void
{
    /*~~~~~
example_config    *base = (example_conf
example_config    *add = (example_conf
example_config    *conf = (example_conf
    /*~~~~~

    conf->enabled = (add->enabled == 0) ? b
    conf->typeOfAction = add->typeOfAction
    strcpy(conf->path, strlen(add->path) ?
    return conf;
}
}
```



We have now looked at how to create simple modules for Apache HTTP Server 2.4 and configuring them. What you do next is entirely up to you, but it is my hope that something valuable has come out of reading this documentation. If you have questions on how to further develop modules, you are welcome to join our [mailing lists](#) or check out the rest of our documentation for further tips.



## Retrieve variables from POST form data

```
typedef struct {
    const char *key;
    const char *value;
} keyValuePair;

keyValuePair *readPost(request_rec *r) {
    apr_array_header_t *pairs = NULL;
    apr_off_t len;
    apr_size_t size;
    int res;
    int i = 0;
    char *buffer;
    keyValuePair *kvp;

    res = ap_parse_form_data(r, NULL, &pairs);
    if (res != OK || !pairs) return NULL;
    kvp = apr_pcalloc(r->pool, sizeof(keyValuePair), 1);
    while (pairs && !apr_is_empty_array(pairs)) {
        ap_form_pair_t *pair = (ap_form_pair_t *) pairs->elts[i];
        apr_brigade_length(pair->value, 1, &len);
        size = (apr_size_t) len;
        buffer = apr_palloc(r->pool, size + 1);
        apr_brigade_flatten(pair->value, buffer, &len);
        buffer[len] = 0;
        kvp[i].key = apr_pstrdup(r->pool, pair->key);
        kvp[i].value = buffer;
        i++;
    }
    return kvp;
}

static int example_handler(request_rec *r)
{
```

```

/*~~~~~*/
keyValuePair *formData;
/*~~~~~*/

formData = readPost(r);
if (formData) {
    int i;
    for (i = 0; &formData[i]; i++) {
        if (formData[i].key && formData[i].value) {
            ap_rprintf(r, "%s = %s\n", formData[i].key, formData[i].value);
        } else if (formData[i].key) {
            ap_rprintf(r, "%s\n", formData[i].key);
        } else if (formData[i].value) {
            ap_rprintf(r, "%s\n", formData[i].value);
        } else {
            break;
        }
    }
}
return OK;
}

```

## Printing out every HTTP header received

```

static int example_handler(request_rec *r)
{
    /*~~~~~*/
    const apr_array_header_t *fields;
    int i;
    apr_table_entry_t *e = 0;
    /*~~~~~*/

    fields = apr_table_elts(r->headers_in);
    e = (apr_table_entry_t *) fields->elts;
    for(i = 0; i < fields->nelts; i++) {

```

```

        ap_rprintf(r, "%s: %s\n", e[i].key,
    }
    return OK;
}

```

## Reading the request body into memory

```

static int util_read(request_rec *r, const
{
    /*~~~~~*/
    int rc = OK;
    /*~~~~~*/

    if((rc = ap_setup_client_block(r, REQUEST_BODY)) != OK)
        return(rc);
}

if(ap_should_client_block(r)) {
    /*~~~~~*/
    char          argsbuffer[HUGE_STRING_LEN];
    apr_off_t     rsize, len_read, rpos;
    apr_off_t     length = r->remaining;
    /*~~~~~*/

    *rbuf = (const char *) apr_pccalloc(r->pool,
    *size = length;
    while((len_read = ap_get_client_block(r, argsbuffer, rpos, rsize)) > 0)
        if((rpos + len_read) > length)
            rsize = length - rpos;
        else {
            rsize = len_read;
        }
}

```

```
        memcpy((char *) *rbuf + rpos, a
              rpos += rsize;
    }
}
return(rc);
}

static int example_handler(request_rec *r)
{
    /*~~~~~*/
    apr_off_t  size;
    const char *buffer;
    /*~~~~~*/

    if(util_read(r, &buffer, &size) == OK)
        ap_rprintf(r, "We read a request bo
    }
    return OK;
}
}
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Redirecting and Remapping with mod\_rewrite

This document supplements the [mod\\_rewrite reference documentation](#). It describes how you can use [mod\\_rewrite](#) to redirect and remap request. This includes many examples of common uses of mod\_rewrite, including detailed descriptions of how each works.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.



## See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Controlling access](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Using RewriteMap](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



## Description:

Assume we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. However, we want that users of the old URL even not recognize that the pages was renamed - that is, we don't want the address to change in their browser.

## Solution:

We rewrite the old URL to the new one internally via the following rule:

```
RewriteEngine on
RewriteRule    "^/foo\.html$"    "/bar.htm
```



**Description:**

Assume again that we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. But this time we want that the users of the old URL get hinted to the new one, i.e. their browsers Location field should change, too.

**Solution:**

We force a HTTP redirect to the new URL which leads to a change of the browsers and thus the users view:

```
RewriteEngine on
RewriteRule    "^/foo\.html$"    "bar.html"
```

**Discussion**

In this example, as contrasted to the [internal](#) example above, we can simply use the Redirect directive. `mod_rewrite` was used in that earlier example in order to hide the redirect from the client:

```
Redirect "/foo.html" "/bar.html"
```



## Description:

If a resource has moved to another server, you may wish to have URLs continue to work for a time on the old server while people update their bookmarks.

## Solution:

You can use `mod_rewrite` to redirect these URLs to the new server, but you might also consider using the `Redirect` or `RedirectMatch` directive.

```
#With mod_rewrite  
RewriteEngine on  
RewriteRule    "^/docs/(.+)"    "http://new
```

```
#With RedirectMatch  
RedirectMatch  "^/docs/(.*)"    "http://new.
```

```
#With Redirect  
Redirect    "/docs/"    "http://new.example.co
```



### Description:

How can we transform a static page `foo.html` into a dynamic variant `foo.cgi` in a seamless way, i.e. without notice by the browser/user.

### Solution:

We just rewrite the URL to the CGI-script and force the handler to be **cgi-script** so that it is executed as a CGI program. This way a request to `/~quux/foo.html` internally leads to the invocation of `/~quux/foo.cgi`.

```
RewriteEngine on
RewriteBase    "/~quux/"
RewriteRule    "^foo\.html$" "foo.cgi"
```



## Description:

How can we make URLs backward compatible (still existing virtually) after migrating document .YYYY to document .XXXX, e.g. after translating a bunch of .html files to .php?

## Solution:

We rewrite the name to its basename and test for existence of the new extension. If it exists, we take that name, else we rewrite the URL to its original state.

```
# backward compatibility ruleset for
# rewriting document.html to document.
# when and only when document.php exists
<Directory "/var/www/html">
    RewriteEngine on
    RewriteBase "/"

    RewriteCond "$1.php" -f
    RewriteCond "$1.html" !-f
    RewriteRule "^(.*)\.html$" "$1.php"
</Directory>
```

## Discussion

This example uses an often-overlooked feature of `mod_rewrite`, by taking advantage of the order of execution of the ruleset. In particular, `mod_rewrite` evaluates the left-hand-side of the `RewriteRule` before it evaluates the `RewriteCond` directives. Consequently, `$1` is already defined by the time the `RewriteCond` directives are evaluated. This allows us to test for the existence of the original (document .html) and target (document .php) files using the same base filename.

This ruleset is designed to use in a per-directory context (In a <Directory> block or in a .htaccess file), so that the -f checks are looking at the correct directory path. You may need to set a RewriteBase directive to specify the directory base that you're working in.



## Description:

The goal of this rule is to force the use of a particular hostname, in preference to other hostnames which may be used to reach the same site. For example, if you wish to force the use of **www.example.com** instead of **example.com**, you might use a variant of the following recipe.

## Solution:

The very best way to solve this doesn't involve `mod_rewrite` at all, but rather uses the `Redirect` directive placed in a virtual host for the non-canonical hostname(s).

```
<VirtualHost *:80>
  ServerName undesired.example.com
  ServerAlias example.com notthis.examp1

  Redirect "/" "http://www.example.com/"
</VirtualHost>

<VirtualHost *:80>
  ServerName www.example.com
</VirtualHost>
```

You can alternatively accomplish this using the `<If>` directive:

```
<If "%{HTTP_HOST} != 'www.example.com'">
  Redirect "/" "http://www.example.com"
</If>
```

Or, for example, to redirect a portion of your site to HTTPS, you might do the following:

```
<If "%{SERVER_PROTOCOL} != 'HTTPS'">
  Redirect "/admin/" "https://www.exam
</If>
```

If, for whatever reason, you still want to use `mod_rewrite` - if, for example, you need this to work with a larger set of `RewriteRules` - you might use one of the recipes below.

For sites running on a port other than 80:

```
RewriteCond "%{HTTP_HOST}"      "!^www\.exa
RewriteCond "%{HTTP_HOST}"      "!^$"
RewriteCond "%{SERVER_PORT}"    "!^80$"
RewriteRule  "^/?(.*)"          "http://www
```

And for a site running on port 80

```
RewriteCond "%{HTTP_HOST}"      "!^www\.exa
RewriteCond "%{HTTP_HOST}"      "!^$"
RewriteRule  "^/?(.*)"          "http://www
```

If you wanted to do this generically for all domain names - that is, if you want to redirect **example.com** to **www.example.com** for all possible values of **example.com**, you could use the following recipe:

```
RewriteCond "%{HTTP_HOST}"      "!^www\." [NC
RewriteCond "%{HTTP_HOST}"      "!^$"
RewriteRule  "^/?(.*)"          "http://www.%
```

These rulesets will work either in your main server configuration file, or in a .htaccess file placed in the DocumentRoot of the server.



### Description:

A particular resource might exist in one of several places, and we want to look in those places for the resource when it is requested. Perhaps we've recently rearranged our directory structure, dividing content into several locations.

### Solution:

The following ruleset searches in two directories to find the resource, and, if not finding it in either place, will attempt to just serve it out of the location requested.

```
RewriteEngine on

# first try to find it in dir1/...
# ...and if found stop and be happy:
RewriteCond          "%{DOCUMENT_ROOT}/di
RewriteRule "^(.+)" "%{DOCUMENT_ROOT}/di

# second try to find it in dir2/...
# ...and if found stop and be happy:
RewriteCond          "%{DOCUMENT_ROOT}/di
RewriteRule "^(.+)" "%{DOCUMENT_ROOT}/di

# else go on for other Alias or Script
# etc.
RewriteRule  "^"  "-"  [PT]
```



## Description:

We have numerous mirrors of our website, and want to redirect people to the one that is located in the country where they are located.

## Solution:

Looking at the hostname of the requesting client, we determine which country they are coming from. If we can't do a lookup on their IP address, we fall back to a default server.

We'll use a [RewriteMap](#) directive to build a list of servers that we wish to use.

```
HostnameLookups on
RewriteEngine on
RewriteMap    multiplex          "txt:/pa
RewriteCond   "%{REMOTE_HOST}"  "([a-z]+
RewriteRule   "^/(.*)$"         "${multiplex:%1
```

```
## map.mirrors -- Multiplexing Map

de http://www.example.de/
uk http://www.example.uk/
com http://www.example.com/
##EOF##
```

## Discussion

This ruleset relies on [HostNameLookups](#) being set on, which can be a significant performance hit.

The [RewriteCond](#) directive captures the last portion of the hostname of the requesting client - the country code - and the

following RewriteRule uses that value to look up the appropriate mirror host in the map file.



## Description:

We wish to provide different content based on the browser, or user-agent, which is requesting the content.

## Solution:

We have to decide, based on the HTTP header "User-Agent", which content to serve. The following config does the following: If the HTTP header "User-Agent" contains "Mozilla/3", the page `foo.html` is rewritten to `foo.NS.html` and the rewriting stops. If the browser is "Lynx" or "Mozilla" of version 1 or 2, the URL becomes `foo.20.html`. All other browsers receive page `foo.32.html`. This is done with the following ruleset:

```
RewriteCond "%{HTTP_USER_AGENT}" "^Mozi
RewriteRule "^foo\.html$" "foo.N

RewriteCond "%{HTTP_USER_AGENT}" "^Lynx
RewriteCond "%{HTTP_USER_AGENT}" "^Mozi
RewriteRule "^foo\.html$" "foo.2

RewriteRule "^foo\.html$" "foo.3
```



## Description:

On some web servers there is more than one URL for a resource. Usually there are canonical URLs (which are actually used and distributed) and those which are just shortcuts, internal ones, and so on. Independent of which URL the user supplied with the request, they should finally see the canonical one in their browser address bar.

## Solution:

We do an external HTTP redirect for all non-canonical URLs to fix them in the location view of the Browser and for all subsequent requests. In the example ruleset below we replace `/puppies` and `/canines` by the canonical `/dogs`.

```
RewriteRule    "^/(puppies|canines)/(.*)"
```

## Discussion:

This should really be accomplished with `Redirect` or `RedirectMatch` directives:

```
RedirectMatch  "^/(puppies|canines)/(.*)"
```



## Description:

Usually the [DocumentRoot](#) of the webserver directly relates to the URL "/". But often this data is not really of top-level priority. For example, you may wish for visitors, on first entering a site, to go to a particular subdirectory /about/. This may be accomplished using the following ruleset:

## Solution:

We redirect the URL / to /about/:

```
RewriteEngine on
RewriteRule    "^/$"    "/about/"    [R]
```

Note that this can also be handled using the [RedirectMatch](#) directive:

```
RedirectMatch  "^/$"  "http://example.com/"
```

Note also that the example rewrites only the root URL. That is, it rewrites a request for `http://example.com/`, but not a request for `http://example.com/page.html`. If you have in fact changed your document root - that is, if **all** of your content is in fact in that subdirectory, it is greatly preferable to simply change your [DocumentRoot](#) directive, or move all of the content up one directory, rather than rewriting URLs.



## Description:

You want a single resource (say, a certain file, like index.php) to handle all requests that come to a particular directory, except those that should go to an existing resource such as an image, or a css file.

## Solution:

As of version 2.2.16, you should use the [FallbackResource](#) directive for this:

```
<Directory "/var/www/my_blog">
  FallbackResource "index.php"
</Directory>
```

However, in earlier versions of Apache, or if your needs are more complicated than this, you can use a variation of the following rewrite set to accomplish the same thing:

```
<Directory "/var/www/my_blog">
  RewriteBase "/my_blog"

  RewriteCond "/var/www/my_blog/%{REQUEST_URI}" !-f
  RewriteCond "/var/www/my_blog/%{REQUEST_URI}" !-d
  RewriteRule "^" "index.php" [PT]
</Directory>
```

If, on the other hand, you wish to pass the requested URI as a query string argument to index.php, you can replace that RewriteRule with:

```
RewriteRule "(.*)" "index.php?$1" [PT, QS]
```

Note that these rulesets can be used in a `.htaccess` file, as well as in a `<Directory>` block.



### Description:

You want to capture a particular value from a query string and either replace it or incorporate it into another component of the URL.

### Solutions:

Many of the solutions in this section will all use the same condition, which leaves the matched value in the %2 backreference. %1 is the beginning of the query string (up to the key of interest), and %3 is the remainder. This condition is a bit complex for flexibility and to avoid double '&&' in the substitutions.

- This solution removes the matching key and value:

```
# Remove mykey=???  
RewriteCond "%{QUERY_STRING}" "(.*(?:mykey=))"  
RewriteRule "(.*)$" "$1%1%3"
```

- This solution uses the captured value in the URL substitution, discarding the rest of the original query by appending a '?':

```
# Copy from query string to PATH_INFO  
RewriteCond "%{QUERY_STRING}" "(.*(?:mykey=))"  
RewriteRule "(.*)$" "$1/products/%2/?"
```

- This solution checks the captured value in a subsequent condition:

```
# Capture the value of mykey in the c  
RewriteCond "%{QUERY_STRING}" "(.*(?:mykey=))"
```

```
RewriteCond "%2" !=not-so-secret-value
RewriteRule "(.*)" - [F]
```

- This solution shows the reverse of the previous ones, copying path components (perhaps PATH\_INFO) from the URL into the query string.

```
# The desired URL might be /products/
# /path?products=kitchen-sink.
RewriteRule "^/?path/([^/]+)/([^/]+)"
```

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

## Using mod\_rewrite to control access

This document supplements the [mod\\_rewrite reference documentation](#). It describes how you can use [mod\\_rewrite](#) to control access to various resources, and other related techniques. This includes many examples of common uses of mod\_rewrite, including detailed descriptions of how each works.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.



### See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Redirection and remapping](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Using RewriteMap](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



### Description:

The following technique forbids the practice of other sites including your images inline in their pages. This practice is often referred to as "hotlinking", and results in your bandwidth being used to serve content for someone else's site.

### Solution:

This technique relies on the value of the HTTP\_REFERER variable, which is optional. As such, it's possible for some people to circumvent this limitation. However, most users will experience the failed request, which should, over time, result in the image being removed from that other site.

There are several ways that you can handle this situation.

In this first example, we simply deny the request, if it didn't initiate from a page on our site. For the purpose of this example, we assume that our site is `www.example.com`.

```
RewriteCond "%{HTTP_REFERER}" "!\^$"
RewriteCond "%{HTTP_REFERER}" "!\www.exam
RewriteRule "\.(gif|jpg|png)$" "-"
```

In this second example, instead of failing the request, we display an alternate image instead.

```
RewriteCond "%{HTTP_REFERER}" "!\^$"
RewriteCond "%{HTTP_REFERER}" "!\www.exam
RewriteRule "\.(gif|jpg|png)$" "/imag
```

In the third example, we redirect the request to an image on some other site.

```
RewriteCond "%{HTTP_REFERER}" "!\^$"
RewriteCond "%{HTTP_REFERER}" "!www.exam
RewriteRule "\.(gif|jpg|png)$" "http://o
```

Of these techniques, the last two tend to be the most effective in getting people to stop hotlinking your images, because they will simply not see the image that they expected to see.

### Discussion:

If all you wish to do is deny access to the resource, rather than redirecting that request elsewhere, this can be accomplished without the use of `mod_rewrite`:

```
SetEnvIf Referer "example\.com" localref
<FilesMatch "\.(jpg|png|gif)$">
    Require env localreferer
</FilesMatch>
```



### Description:

In this recipe, we discuss how to block persistent requests from a particular robot, or user agent.

The standard for robot exclusion defines a file, `/robots.txt` that specifies those portions of your website where you wish to exclude robots. However, some robots do not honor these files.

Note that there are methods of accomplishing this which do not use `mod_rewrite`. Note also that any technique that relies on the clients `USER_AGENT` string can be circumvented very easily, since that string can be changed.

### Solution:

We use a ruleset that specifies the directory to be protected, and the client `USER_AGENT` that identifies the malicious or persistent robot.

In this example, we are blocking a robot called `NameOfBadRobot` from a location `/secret/files`. You may also specify an IP address range, if you are trying to block that user agent only from the particular source.

```
RewriteCond "%{HTTP_USER_AGENT}"    "^Nam
RewriteCond "%{REMOTE_ADDR}"        "=123
RewriteRule  "^/secret/files/"      "- "   [F
```

### Discussion:

Rather than using `mod_rewrite` for this, you can accomplish the same end using alternate means, as illustrated here:

```
SetEnvIfNoCase User-Agent "^NameOfBadRob  
<Location "/secret/files">  
  <RequireAll>  
    Require all granted  
    Require not env goaway  
  </RequireAll>  
</Location>
```

As noted above, this technique is trivial to circumvent, by simply modifying the USER\_AGENT request header. If you are experiencing a sustained attack, you should consider blocking it at a higher level, such as at your firewall.



## Description:

We wish to maintain a blacklist of hosts, rather like `hosts.deny`, and have those hosts blocked from accessing our server.

## Solution:

```
RewriteEngine on
RewriteMap      hosts-deny  "txt:/path/to/
RewriteCond     "${hosts-deny:%{REMOTE_ADD
RewriteCond     "${hosts-deny:%{REMOTE_HOS
RewriteRule     "^"  "-"  [F]
```

```
##
## hosts.deny
##
## ATTENTION! This is a map, not a list, even when we
## treat it as such.
## mod_rewrite parses it for key/value pairs, so at least
## a
## dummy value "-" must be present for each entry.
##

193.102.180.41 -
bsdti1.sdm.de -
192.76.162.40 -
```

## Discussion:

The second `RewriteCond` assumes that you have `HostNameLookups` turned on, so that client IP addresses will be resolved. If that's not the case, you should drop the second `RewriteCond`, and drop the `[OR]` flag from the first `RewriteCond`.



## Description:

Redirect requests based on the Referer from which the request came, with different targets per Referer.

## Solution:

The following ruleset uses a map file to associate each Referer with a redirection target.

```
RewriteMap deflector "txt:/path/to/defl

RewriteCond "%{HTTP_REFERER}" !=""
RewriteCond "${deflector:%{HTTP_REFERER}}
RewriteRule "^" "%{HTTP_REFERER}" [R,L]

RewriteCond "%{HTTP_REFERER}" !=""
RewriteCond "${deflector:%{HTTP_REFERER}}
RewriteRule "^" "${deflector:%{HTTP_REFERER}}
```

The map file lists redirection targets for each referer, or, if we just wish to redirect back to where they came from, a "-" is placed in the map:

```
##
## deflector.map
##

http://badguys.example.com/bad/index.htm
http://badguys.example.com/bad/index2.ht
http://badguys.example.com/bad/index3.ht
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Dynamic mass virtual hosts with mod\_rewrite

This document supplements the [mod\\_rewrite reference documentation](#). It describes how you can use [mod\\_rewrite](#) to create dynamically configured virtual hosts.

mod\_rewrite is not the best way to configure virtual hosts. You should first consider the [alternatives](#) before resorting to mod\_rewrite. See also the "[how to avoid mod\\_rewrite](#) document.



## See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Redirection and remapping](#)
- [Controlling access](#)
- [Proxying](#)
- [RewriteMap](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



## Description:

We want to automatically create a virtual host for every hostname which resolves in our domain, without having to create new VirtualHost sections.

In this recipe, we assume that we'll be using the hostname `www.SITE.example.com` for each user, and serve their content out of `/home/SITE/www`.

## Solution:

```
RewriteEngine on

RewriteMap    lowercase int:tolower

RewriteCond   "${lowercase:%{HTTP_HOST}}"
RewriteRule   "^(.*)" "/home/%1/www$1"
```

## Discussion

You will need to take care of the DNS resolution - Apache does not handle name resolution. You'll need either to create CNAME records for each hostname, or a DNS wildcard record. Creating DNS records is beyond the scope of this document.

The internal `tolower` RewriteMap directive is used to ensure that the hostnames being used are all lowercase, so that there is no ambiguity in the directory structure which must be created.

Parentheses used in a [RewriteCond](#) are captured into the

backreferences %1, %2, etc, while parentheses used in RewriteRule are captured into the backreferences \$1, \$2, etc.

As with many techniques discussed in this document, `mod_rewrite` really isn't the best way to accomplish this task. You should, instead, consider using `mod_vhost_alias` instead, as it will much more gracefully handle anything beyond serving static files, such as any dynamic content, and Alias resolution.



This extract from `httpd.conf` does the same thing as [the first example](#). The first half is very similar to the corresponding part above, except for some changes, required for backward compatibility and to make the `mod_rewrite` part work properly; the second half configures `mod_rewrite` to do the actual work.

Because `mod_rewrite` runs before other URI translation modules (e.g., `mod_alias`), `mod_rewrite` must be told to explicitly ignore any URLs that would have been handled by those modules. And, because these rules would otherwise bypass any `ScriptAlias` directives, we must have `mod_rewrite` explicitly enact those mappings.

```
# get the server name from the Host: header
UseCanonicalName Off

# splittable logs
LogFormat "%{Host}i %h %l %u %t \"%r\" %s %>
CustomLog "logs/access_log" vcommon

<Directory "/www/hosts">
    # ExecCGI is needed here because we can
    # CGI execution in the way that ScriptAl
    Options FollowSymLinks ExecCGI
</Directory>

RewriteEngine On

# a ServerName derived from a Host: header r
RewriteMap lowercase int:tolower

## deal with normal documents first:
# allow Alias "/icons/" to work - repeat for
RewriteCond "%{REQUEST_URI}" "!^/icons/"
# allow CGIs to work
```

```
RewriteCond "%{REQUEST_URI}" "!^/cgi-bin/"
# do the magic
RewriteRule "^/(.*)$" "/www/hosts/${lowerc

## and now deal with CGIs - we have to force
RewriteCond "%{REQUEST_URI}" "^/cgi-bin/"
RewriteRule "^/(.*)$" "/www/hosts/${lowerc
```



This arrangement uses more advanced `mod_rewrite` features to work out the translation from virtual host to document root, from a separate configuration file. This provides more flexibility, but requires more complicated configuration.

The `vhost.map` file should look something like this:

```
customer-1.example.com /www/customers/1
customer-2.example.com /www/customers/2
# ...
customer-N.example.com /www/customers/N
```

The `httpd.conf` should contain the following:

```
RewriteEngine on

RewriteMap lowercase int:tolower

# define the map file
RewriteMap vhost "txt:/www/conf/vhost

# deal with aliases as above
RewriteCond "%{REQUEST_URI}"
RewriteCond "%{REQUEST_URI}"
RewriteCond "${lowercase:%{SERVER_NAME}}"
# this does the file-based remap
RewriteCond "${vhost:%1}"
RewriteRule "^/(.*)$"

RewriteCond "%{REQUEST_URI}"
RewriteCond "${lowercase:%{SERVER_NAME}}"
RewriteCond "${vhost:%1}"
RewriteRule "^/cgi-bin/(.*)$"


```

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Using mod\_rewrite for Proxying

This document supplements the [mod\\_rewrite reference documentation](#). It describes how to use the RewriteRule's [P] flag to proxy content to another server. A number of recipes are provided that describe common scenarios.



## See also

[Module documentation](#)

[mod\\_rewrite introduction](#)

[Redirection and remapping](#)

[Controlling access](#)

[Virtual hosts](#)

[Using RewriteMap](#)

[Advanced techniques](#)

[When not to use mod\\_rewrite](#)



## Description:

mod\_rewrite provides the [P] flag, which allows URLs to be passed, via mod\_proxy, to another server. Two examples are given here. In one example, a URL is passed directly to another server, and served as though it were a local URL. In the other example, we proxy missing content to a back-end server.

## Solution:

To simply map a URL to another server, we use the [P] flag, as follows:

```
RewriteEngine on
RewriteBase "/products/"
RewriteRule "^widget/(.*)$" "http://
ProxyPassReverse "/products/widget/" "ht
```

In the second example, we proxy the request only if we can't find the resource locally. This can be very useful when you're migrating from one server to another, and you're not sure if all the content has been migrated yet.

```
RewriteCond "%{REQUEST_FILENAME}"
RewriteCond "%{REQUEST_FILENAME}"
RewriteRule "^/(.*)" "http://old.example
ProxyPassReverse "/" "http://old.example
```

## Discussion:

In each case, we add a [ProxyPassReverse](#) directive to ensure that any redirects issued by the backend are correctly passed on to the client.

Consider using either [ProxyPass](#) or [ProxyPassMatch](#) whenever possible in preference to `mod_rewrite`.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Advanced Techniques with mod\_rewrite

This document supplements the [mod\\_rewrite reference documentation](#). It provides a few advanced techniques using mod\_rewrite.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.



## See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Redirection and remapping](#)
- [Controlling access](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Using RewriteMap](#)
- [When not to use mod\\_rewrite](#)



## Description:

A common technique for distributing the burden of server load or storage space is called "sharding". When using this method, a front-end server will use the url to consistently "shard" users or objects to separate backend servers.

## Solution:

A mapping is maintained, from users to target servers, in external map files. They look like:

```
user1 physical_host_of_user1
user2 physical_host_of_user2
: :
```

We put this into a map .users-to-hosts file. The aim is to map;

```
/u/user1/anypath
```

to

```
http://physical_host_of_user1/u/user/anypath
```

thus every URL path need not be valid on every backend physical host. The following ruleset does this for us with the help of the map files assuming that server0 is a default server which will be used if a user has no entry in the map:

```
RewriteEngine on
RewriteMap      users-to-hosts      "txt:/p
RewriteRule     "^/u/([^/]+)/?(.*)"
```

See the [RewriteMap](#) documentation for more discussion of the

syntax of this directive.



## Description:

We wish to dynamically generate content, but store it statically once it is generated. This rule will check for the existence of the static file, and if it's not there, generate it. The static files can be removed periodically, if desired (say, via cron) and will be regenerated on demand.

## Solution:

This is done via the following ruleset:

```
# This example is valid in per-directory
RewriteCond "%{REQUEST_URI}"    "-U"
RewriteRule  "^(.+)\.html$"      "/re
```

The `-U` operator determines whether the test string (in this case, `REQUEST_URI`) is a valid URL. It does this via a subrequest. In the event that this subrequest fails - that is, the requested resource doesn't exist - this rule invokes the CGI program `/regenerate_page.cgi`, which generates the requested resource and saves it into the document directory, so that the next time it is requested, a static copy can be served.

In this way, documents that are infrequently updated can be served in static form. If documents need to be refreshed, they can be deleted from the document directory, and they will then be regenerated the next time they are requested.



## Description:

We wish to randomly distribute load across several servers using `mod_rewrite`.

## Solution:

We'll use [RewriteMap](#) and a list of servers to accomplish this.

```
RewriteEngine on
RewriteMap lb "rnd:/path/to/serverlist.t
RewriteRule "^/(.*)" "http://${lb:server
```

`serverlist.txt` will contain a list of the servers:

```
## serverlist.txt

servers one.example.com|two.example.com|three.example.com
```

If you want one particular server to get more of the load than the others, add it more times to the list.

## Discussion

Apache comes with a load-balancing module - [mod\\_proxy\\_balancer](#) - which is far more flexible and featureful than anything you can cobble together using `mod_rewrite`.



## Description:

Some sites with thousands of users use a structured homedir layout, *i.e.* each homedir is in a subdirectory which begins (for instance) with the first character of the username. So,

`/~larry/anypath` is

`/home/l/larry/public_html/anypath` while

`/~waldo/anypath` is

`/home/w/waldo/public_html/anypath`.

## Solution:

We use the following ruleset to expand the tilde URLs into the above layout.

```
RewriteEngine on
RewriteRule    "^/~(( [a-z] ) [a-z0-9]+) (.*)"

```



**Description:**

By default, redirecting to an HTML anchor doesn't work, because `mod_rewrite` escapes the `#` character, turning it into `%23`. This, in turn, breaks the redirection.

**Solution:**

Use the `[NE]` flag on the `RewriteRule`. `NE` stands for No Escape.

**Discussion:**

This technique will of course also work with other special characters that `mod_rewrite`, by default, URL-encodes.



### Description:

We wish to use `mod_rewrite` to serve different content based on the time of day.

### Solution:

There are a lot of variables named `TIME_XXX` for rewrite conditions. In conjunction with the special lexicographic comparison patterns `<STRING`, `>STRING` and `=STRING` we can do time-dependent redirects:

```
RewriteEngine on
RewriteCond    "%{TIME_HOUR}%{TIME_MIN}"
RewriteCond    "%{TIME_HOUR}%{TIME_MIN}"
RewriteRule    "^foo\.html$"
RewriteRule    "^foo\.html$"
```

This provides the content of `foo.day.html` under the URL `foo.html` from `07:01-18:59` and at the remaining time the contents of `foo.night.html`.

[mod\\_cache](#), intermediate proxies and browsers may each cache responses and cause the either page to be shown outside of the time-window configured. [mod\\_expires](#) may be used to control this effect. You are, of course, much better off simply serving the content dynamically, and customizing it based on the time of day.



## Description:

At time, we want to maintain some kind of status when we perform a rewrite. For example, you want to make a note that you've done that rewrite, so that you can check later to see if a request can via that rewrite. One way to do this is by setting an environment variable.

## Solution:

Use the [E] flag to set an environment variable.

```
RewriteEngine on
RewriteRule    "^/horse/(.*)"    "/pony/$1
```

Later in your ruleset you might check for this environment variable using a RewriteCond:

```
RewriteCond    "%{ENV:rewritten}"    "=1"
```

Note that environment variables do not survive an external redirect. You might consider using the [CO] flag to set a cookie.



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

## When not to use mod\_rewrite

This document supplements the [mod\\_rewrite reference documentation](#). It describes perhaps one of the most important concepts about [mod\\_rewrite](#) - namely, when to avoid using it.

[mod\\_rewrite](#) should be considered a last resort, when other alternatives are found wanting. Using it when there are simpler alternatives leads to configurations which are confusing, fragile, and hard to maintain. Understanding what other alternatives are available is a very important step towards [mod\\_rewrite](#) mastery.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.

The most common situation in which [mod\\_rewrite](#) is the right tool is when the very best solution requires access to the server configuration files, and you don't have that access. Some configuration directives are only available in the server configuration file. So if you are in a hosting situation where you only have .htaccess files to work with, you may need to resort to [mod\\_rewrite](#).



### See also

[Module documentation](#)

[mod\\_rewrite introduction](#)

[Redirection and remapping](#)

[Controlling access](#)

[Virtual hosts](#)

[Proxying](#)

[Using RewriteMap](#)

[Advanced techniques](#)



`mod_alias` provides the `Redirect` and `RedirectMatch` directives, which provide a means to redirect one URL to another. This kind of simple redirection of one URL, or a class of URLs, to somewhere else, should be accomplished using these directives rather than `RewriteRule`. `RedirectMatch` allows you to include a regular expression in your redirection criteria, providing many of the benefits of using `RewriteRule`.

A common use for `RewriteRule` is to redirect an entire class of URLs. For example, all URLs in the `/one` directory must be redirected to `http://one.example.com/`, or perhaps all `http` requests must be redirected to `https`.

These situations are better handled by the `Redirect` directive. Remember that `Redirect` preserves path information. That is to say, a redirect for a URL `/one` will also redirect all URLs under that, such as `/one/two.html` and `/one/three/four.html`.

To redirect URLs under `/one` to `http://one.example.com`, do the following:

```
Redirect "/one/" "http://one.example.com/"
```

To redirect one hostname to another, for example `example.com` to `www.example.com`, see the [Canonical Hostnames](#) recipe.

To redirect `http` URLs to `https`, do the following:

```
<VirtualHost *:80>
    ServerName www.example.com
    Redirect "/" "https://www.example.com/"
</VirtualHost>
```

```
<VirtualHost *:443>
  ServerName www.example.com
  # ... SSL configuration goes here
</VirtualHost>
```

The use of `RewriteRule` to perform this task may be appropriate if there are other `RewriteRule` directives in the same scope. This is because, when there are `Redirect` and `RewriteRule` directives in the same scope, the `RewriteRule` directives will run first, regardless of the order of appearance in the configuration file.

In the case of the *http-to-https* redirection, the use of `RewriteRule` would be appropriate if you don't have access to the main server configuration file, and are obliged to perform this task in a `.htaccess` file instead.



The [Alias](#) directive provides mapping from a URI to a directory - usually a directory outside of your [DocumentRoot](#). Although it is possible to perform this mapping with [mod\\_rewrite](#), [Alias](#) is the preferred method, for reasons of simplicity and performance.

### Using Alias

```
Alias "/cats" "/var/www/virtualhosts/felines/htdocs"
```

The use of [mod\\_rewrite](#) to perform this mapping may be appropriate when you do not have access to the server configuration files. Alias may only be used in server or virtualhost context, and not in a `.htaccess` file.

Symbolic links would be another way to accomplish the same thing, if you have `Options FollowSymLinks` enabled on your server.



## Virtual Hosting

Although it is possible to handle [virtual hosts with mod\\_rewrite](#), it is seldom the right way. Creating individual `<VirtualHost>` blocks is almost always the right way to go. In the event that you have an enormous number of virtual hosts, consider using [mod\\_vhost\\_alias](#) to create these hosts automatically.

Modules such as [mod\\_macro](#) are also useful for creating a large number of virtual hosts dynamically.

Using [mod\\_rewrite](#) for virtualhost creation may be appropriate if you are using a hosting service that does not provide you access to the server configuration files, and you are therefore restricted to configuration using `.htaccess` files.

See the [virtual hosts with mod\\_rewrite](#) document for more details on how you might accomplish this if it still seems like the right approach.



## Simple Proxying

[RewriteRule](#) provides the [\[P\]](#) flag to pass rewritten URIs through [mod\\_proxy](#).

```
RewriteRule "^/?images(.*)" "http://imageser
```

However, in many cases, when there is no actual pattern matching needed, as in the example shown above, the [ProxyPass](#) directive is a better choice. The example here could be rendered as:

```
ProxyPass "/images/" "http://imageserver.loc
```

Note that whether you use [RewriteRule](#) or [ProxyPass](#), you'll still need to use the [ProxyPassReverse](#) directive to catch redirects issued from the back-end server:

```
ProxyPassReverse "/images/" "http://imageser
```

You may need to use `RewriteRule` instead when there are other `RewriteRules` in effect in the same scope, as a `RewriteRule` will usually take effect before a `ProxyPass`, and so may preempt what you're trying to accomplish.



## Environment Variable Testing

[mod\\_rewrite](#) is frequently used to take a particular action based on the presence or absence of a particular environment variable or request header. This can be done more efficiently using the [<If>](#).

Consider, for example, the common scenario where [RewriteRule](#) is used to enforce a canonical hostname, such as `www.example.com` instead of `example.com`. This can be done using the [<If>](#) directive, as shown here:

```
<If "req('Host') != 'www.example.com'">
    Redirect "/" "http://www.example.com/"
</If>
```

This technique can be used to take actions based on any request header, response header, or environment variable, replacing [mod\\_rewrite](#) in many common scenarios.

See especially the [expression evaluation documentation](#) for a overview of what types of expressions you can use in [<If>](#) sections, and in certain other directives.



# APACHE

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Rewrite](#)

# Apache mod\_rewrite Technical Details

This document discusses some of the technical details of mod\_rewrite and URL matching.



## See also

- [Module documentation](#)
- [mod\\_rewrite introduction](#)
- [Redirection and remapping](#)
- [Controlling access](#)
- [Virtual hosts](#)
- [Proxying](#)
- [Using RewriteMap](#)
- [Advanced techniques](#)
- [When not to use mod\\_rewrite](#)



## URL PHASES

The Apache HTTP Server handles requests in several phases. At each of these phases, one or more modules may be called upon to handle that portion of the request lifecycle. Phases include things like URL-to-filename translation, authentication, authorization, content, and logging. (This is not an exhaustive list.)

`mod_rewrite` acts in two of these phases (or "hooks", as they are often called) to influence how URLs may be rewritten.

First, it uses the URL-to-filename translation hook, which occurs after the HTTP request has been read, but before any authorization starts. Secondly, it uses the Fixup hook, which is after the authorization phases, and after per-directory configuration files (`.htaccess` files) have been read, but before the content handler is called.

So, after a request comes in and a corresponding server or virtual host has been determined, the rewriting engine starts processing any `mod_rewrite` directives appearing in the per-server configuration. (i.e., in the main server configuration file and `<Virtualhost>` sections.) This happens in the URL-to-filename phase.

A few steps later, once the final data directories have been found, the per-directory configuration directives (`.htaccess` files and `<Directory>` blocks) are applied. This happens in the Fixup phase.

In each of these cases, `mod_rewrite` rewrites the `REQUEST_URI` either to a new URL, or to a filename.

In per-directory context (i.e., within `.htaccess` files and `Directory` blocks), these rules are being applied after a URL has already been translated to a filename. Because of this, the URL-

path that `mod_rewrite` initially compares [RewriteRule](#) directives against is the full filesystem path to the translated filename with the current directories path (including a trailing slash) removed from the front.

To illustrate: If rules are in `/var/www/foo/.htaccess` and a request for `/foo/bar/baz` is being processed, an expression like `^bar/baz$` would match.

If a substitution is made in per-directory context, a new internal subrequest is issued with the new URL, which restarts processing of the request phases. If the substitution is a relative path, the [RewriteBase](#) directive determines the URL-path prefix prepended to the substitution. In per-directory context, care must be taken to create rules which will eventually (in some future "round" of per-directory rewrite processing) not perform a substitution to avoid looping. (See [RewriteLooping](#) for further discussion of this problem.)

Because of this further manipulation of the URL in per-directory context, you'll need to take care to craft your rewrite rules differently in that context. In particular, remember that the leading directory path will be stripped off of the URL that your rewrite rules will see. Consider the examples below for further clarification.

Location of rule	Rule
VirtualHost section	<code>RewriteRule "^/images/(+)\.jpg" "/images/\$1.gif"</code>
.htaccess file in document root	<code>RewriteRule "^images/(+)\.jpg" "images/\$1.gif"</code>
.htaccess file in images directory	<code>RewriteRule "^(+)\.jpg" "\$1.gif"</code>

For even more insight into how `mod_rewrite` manipulates URLs in

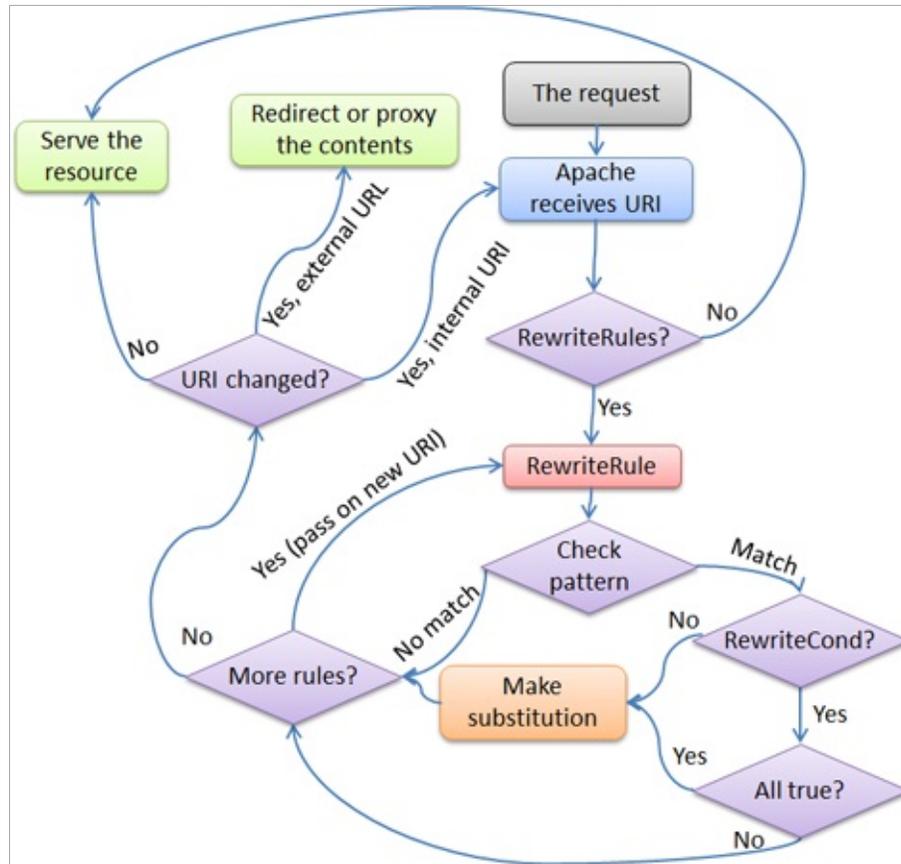
different contexts, you should consult the [log entries](#) made during rewriting.



## URL Rewriting

Now when `mod_rewrite` is triggered in these two API phases, it reads the configured rulesets from its configuration structure (which itself was either created on startup for per-server context or during the directory walk of the Apache kernel for per-directory context). Then the URL rewriting engine is started with the contained ruleset (one or more rules together with their conditions). The operation of the URL rewriting engine itself is exactly the same for both configuration contexts. Only the final result processing is different.

The order of rules in the ruleset is important because the rewriting engine processes them in a special (and not very obvious) order. The rule is this: The rewriting engine loops through the ruleset rule by rule (`RewriteRule` directives) and when a particular rule matches it optionally loops through existing corresponding conditions (`RewriteCond` directives). For historical reasons the conditions are given first, and so the control flow is a little bit long-winded. See Figure 1 for more details.



**Figure**

**1:** The control flow through the rewriting ruleset

First the URL is matched against the *Pattern* of each rule. If it fails, `mod_rewrite` immediately stops processing this rule, and continues with the next rule. If the *Pattern* matches, `mod_rewrite` looks for corresponding rule conditions (`RewriteCond` directives, appearing immediately above the `RewriteRule` in the configuration). If none are present, it substitutes the URL with a new value, which is constructed from the string *Substitution*, and goes on with its rule-looping. But if conditions exist, it starts an inner loop for processing them in the order that they are listed. For conditions, the logic is different: we don't match a pattern against the current URL. Instead we first create a string *TestString* by expanding variables, back-references, map lookups, etc. and then we try to match *CondPattern* against it. If the pattern doesn't match, the complete set of conditions and the corresponding rule fails. If the pattern matches, then the next condition is processed until no

more conditions are available. If all conditions match, processing is continued with the substitution of the URL with *Substitution*.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

## Guide to writing output filters

There are a number of common pitfalls encountered when writing output filters; this page aims to document best practice for authors of new or existing filters.

This document is applicable to both version 2.0 and version 2.2 of the Apache HTTP Server; it specifically targets RESOURCE-level or CONTENT\_SET-level filters though some advice is generic to all types of filter.



## Filtering Buckets

Each time a filter is invoked, it is passed a *bucket brigade*, containing a sequence of *buckets* which represent both data content and metadata. Every bucket has a *bucket type*; a number of bucket types are defined and used by the `httpd` core modules (and the `apr-util` library which provides the bucket brigade interface), but modules are free to define their own types.

Output filters must be prepared to process buckets of non-standard types; with a few exceptions, a filter need not care about the types of buckets being filtered.

A filter can tell whether a bucket represents either data or metadata using the `APR_BUCKET_IS_METADATA` macro. Generally, all metadata buckets should be passed down the filter chain by an output filter. Filters may transform, delete, and insert data buckets as appropriate.

There are two metadata bucket types which all filters must pay attention to: the `EOS` bucket type, and the `FLUSH` bucket type. An `EOS` bucket indicates that the end of the response has been reached and no further buckets need be processed. A `FLUSH` bucket indicates that the filter should flush any buffered buckets (if applicable) down the filter chain immediately.

`FLUSH` buckets are sent when the content generator (or an upstream filter) knows that there may be a delay before more content can be sent. By passing `FLUSH` buckets down the filter chain immediately, filters ensure that the client is not kept waiting for pending data longer than necessary.

Filters can create `FLUSH` buckets and pass these down the filter chain if desired. Generating `FLUSH` buckets unnecessarily, or too frequently, can harm network utilisation since it may force large

numbers of small packets to be sent, rather than a small number of larger packets. The section on [Non-blocking bucket reads](#) covers a case where filters are encouraged to generate FLUSH buckets.

### **Example bucket brigade**

```
HEAP FLUSH FILE EOS
```

This shows a bucket brigade which may be passed to a filter; it contains two metadata buckets (FLUSH and EOS), and two data buckets (HEAP and FILE).



For any given request, an output filter might be invoked only once and be given a single brigade representing the entire response. It is also possible that the number of times a filter is invoked for a single response is proportional to the size of the content being filtered, with the filter being passed a brigade containing a single bucket each time. Filters must operate correctly in either case.

An output filter which allocates long-lived memory every time it is invoked may consume memory proportional to response size. Output filters which need to allocate memory should do so once per response; see [Maintaining state](#) below.

An output filter can distinguish the final invocation for a given response by the presence of an EOS bucket in the brigade. Any buckets in the brigade after an EOS should be ignored.

An output filter should never pass an empty brigade down the filter chain. To be defensive, filters should be prepared to accept an empty brigade, and should return success without passing this brigade on down the filter chain. The handling of an empty brigade should have no side effects (such as changing any state private to the filter).

### How to handle an empty brigade

```
apr_status_t dummy_filter(ap_filter_t *f, apr_bucket_brigade *b)
{
    if (APR_BRIGADE_EMPTY(bb)) {
        return APR_SUCCESS;
    }
    ...
}
```



## Bucket structure

A bucket brigade is a doubly-linked list of buckets. The list is terminated (at both ends) by a *sentinel* which can be distinguished from a normal bucket by comparing it with the pointer returned by `APR_BRIGADE_SENTINEL`. The list sentinel is in fact not a valid bucket structure; any attempt to call normal bucket functions (such as `apr_bucket_read`) on the sentinel will have undefined behaviour (i.e. will crash the process).

There are a variety of functions and macros for traversing and manipulating bucket brigades; see the [apr\\_buckets.h](#) header for complete coverage. Commonly used macros include:

**`APR_BRIGADE_FIRST(bb)`**

returns the first bucket in brigade `bb`

**`APR_BRIGADE_LAST(bb)`**

returns the last bucket in brigade `bb`

**`APR_BUCKET_NEXT(e)`**

gives the next bucket after bucket `e`

**`APR_BUCKET_PREV(e)`**

gives the bucket before bucket `e`

The `apr_bucket_brigade` structure itself is allocated out of a pool, so if a filter creates a new brigade, it must ensure that memory use is correctly bounded. A filter which allocates a new brigade out of the request pool (`r->pool`) on every invocation, for example, will fall foul of the [warning above](#) concerning memory use. Such a filter should instead create a brigade on the first invocation per request, and store that brigade in its [state structure](#).

It is generally never advisable to use `apr_brigade_destroy` to "destroy" a brigade unless you know for certain that the brigade will never be used again, even then, it should be used

rarely. The memory used by the brigade structure will not be released by calling this function (since it comes from a pool), but the associated pool cleanup is unregistered. Using `apr_brigade_destroy` can in fact cause memory leaks; if a "destroyed" brigade contains buckets when its containing pool is destroyed, those buckets will *not* be immediately destroyed.

In general, filters should use `apr_brigade_cleanup` in preference to `apr_brigade_destroy`.



## Processing buckets

When dealing with non-metadata buckets, it is important to understand that the "apr\_bucket \*" object is an abstract *representation of data*:

1. The amount of data represented by the bucket may or may not have a determinate length; for a bucket which represents data of indeterminate length, the `->length` field is set to the value `(apr_size_t)-1`. For example, buckets of the PIPE bucket type have an indeterminate length; they represent the output from a pipe.
2. The data represented by a bucket may or may not be mapped into memory. The FILE bucket type, for example, represents data stored in a file on disk.

Filters read the data from a bucket using the `apr_bucket_read` function. When this function is invoked, the bucket may *morph* into a different bucket type, and may also insert a new bucket into the bucket brigade. This must happen for buckets which represent data not mapped into memory.

To give an example; consider a bucket brigade containing a single FILE bucket representing an entire file, 24 kilobytes in size:

```
FILE(0K-24K)
```

When this bucket is read, it will read a block of data from the file, morph into a HEAP bucket to represent that data, and return the data to the caller. It also inserts a new FILE bucket representing the remainder of the file; after the `apr_bucket_read` call, the brigade looks like:

```
HEAP(8K) FILE(8K-24K)
```



## Filtering brigades

The basic function of any output filter will be to iterate through the passed-in brigade and transform (or simply examine) the content in some manner. The implementation of the iteration loop is critical to producing a well-behaved output filter.

Taking an example which loops through the entire brigade as follows:

### Bad output filter -- do not imitate!

```
apr_bucket *e = APR_BRIGADE_FIRST(bb);
const char *data;
apr_size_t length;

while (e != APR_BRIGADE_SENTINEL(bb)) {
    apr_bucket_read(e, &data, &length, APR_BLOCK_READ);
    e = APR_BUCKET_NEXT(e);
}

return ap_pass_brigade(bb);
```

The above implementation would consume memory proportional to content size. If passed a FILE bucket, for example, the entire file contents would be read into memory as each `apr_bucket_read` call morphed a FILE bucket into a HEAP bucket.

In contrast, the implementation below will consume a fixed amount of memory to filter any brigade; a temporary brigade is needed and must be allocated only once per response, see the [Maintaining state](#) section.

### Better output filter

```
apr_bucket *e;
const char *data;
apr_size_t length;

while ((e = APR_BRIGADE_FIRST(bb)) != APR_BRIGADE_SENTINEL(bb))
```

```
rv = apr_bucket_read(e, &data, &length, APR_BLOCK_READ);
if (rv) ...;
/* Remove bucket e from bb. */
APR_BUCKET_REMOVE(e);
/* Insert it into temporary brigade. */
APR_BRIGADE_INSERT_HEAD(tmpbb, e);
/* Pass brigade downstream. */
rv = ap_pass_brigade(f->next, tmpbb);
if (rv) ...;
apr_brigade_cleanup(tmpbb);
```

```
}
```



## Maintaining state

A filter which needs to maintain state over multiple invocations per response can use the `->ctx` field of its `ap_filter_t` structure. It is typical to store a temporary brigade in such a structure, to avoid having to allocate a new brigade per invocation as described in the [Brigade structure](#) section.

### Example code to maintain filter state

```
struct dummy_state {
    apr_bucket_brigade *tmpbb;
    int filter_state;
    ...
};

apr_status_t dummy_filter(ap_filter_t *f, apr_bucket_brigade *b
{
    struct dummy_state *state;

    state = f->ctx;
    if (state == NULL) {

        /* First invocation for this response: initialise state
        */
        f->ctx = state = apr_palloc(f->r->pool, sizeof *state);

        state->tmpbb = apr_brigade_create(f->r->pool, f->c->buc
        state->filter_state = ...;
    }
    ...
}
```



## Managing Buckets

If a filter decides to store buckets beyond the duration of a single filter function invocation (for example storing them in its `->ctx` state structure), those buckets must be *set aside*. This is necessary because some bucket types provide buckets which represent temporary resources (such as stack memory) which will fall out of scope as soon as the filter chain completes processing the brigade.

To setaside a bucket, the `apr_bucket_setaside` function can be called. Not all bucket types can be setaside, but if successful, the bucket will have morphed to ensure it has a lifetime at least as long as the pool given as an argument to the `apr_bucket_setaside` function.

Alternatively, the `ap_save_brigade` function can be used, which will move all the buckets into a separate brigade containing buckets with a lifetime as long as the given pool argument. This function must be used with care, taking into account the following points:

1. On return, `ap_save_brigade` guarantees that all the buckets in the returned brigade will represent data mapped into memory. If given an input brigade containing, for example, a PIPE bucket, `ap_save_brigade` will consume an arbitrary amount of memory to store the entire output of the pipe.
2. When `ap_save_brigade` reads from buckets which cannot be setaside, it will always perform blocking reads, removing the opportunity to use [Non-blocking bucket reads](#).
3. If `ap_save_brigade` is used without passing a non-NULL "saveto" (destination) brigade parameter, the function will create a new brigade, which may cause memory use to be

proportional to content size as described in the [Brigade structure](#) section.

Filters must ensure that any buffered data is processed and passed down the filter chain during the last invocation for a given response (a brigade containing an EOS bucket). Otherwise such data will be lost.



The `apr_bucket_read` function takes an `apr_read_type_e` argument which determines whether a *blocking* or *non-blocking* read will be performed from the data source. A good filter will first attempt to read from every data bucket using a non-blocking read; if that fails with `APR_EAGAIN`, then send a FLUSH bucket down the filter chain, and retry using a blocking read.

This mode of operation ensures that any filters further down the filter chain will flush any buffered buckets if a slow content source is being used.

A CGI script is an example of a slow content source which is implemented as a bucket type. `mod_cgi` will send PIPE buckets which represent the output from a CGI script; reading from such a bucket will block when waiting for the CGI script to produce more output.

### Example code using non-blocking bucket reads

```
apr_bucket *e;
apr_read_type_e mode = APR_NONBLOCK_READ;

while ((e = APR_BRIGADE_FIRST(bb)) != APR_BRIGADE_SENTINEL(bb))
    apr_status_t rv;

    rv = apr_bucket_read(e, &data, &length, mode);
    if (rv == APR_EAGAIN && mode == APR_NONBLOCK_READ) {

        /* Pass down a brigade containing a flush bucket: */
        APR_BRIGADE_INSERT_TAIL(tmpbb, apr_bucket_flush_create(
        rv = ap_pass_brigade(f->next, tmpbb);
        apr_brigade_cleanup(tmpbb);
        if (rv != APR_SUCCESS) return rv;

        /* Retry, using a blocking read. */
        mode = APR_BLOCK_READ;
        continue;
    }
    else if (rv != APR_SUCCESS) {
        /* handle errors */
    }
}
```

```
/* Next time, try a non-blocking read first. */  
mode = APR_NONBLOCK_READ;  
...  
}
```



~~Rules for output filters~~

In summary, here is a set of rules for all output filters to follow:

1. Output filters should not pass empty brigades down the filter chain, but should be tolerant of being passed empty brigades.
  2. Output filters must pass all metadata buckets down the filter chain; FLUSH buckets should be respected by passing any pending or buffered buckets down the filter chain.
  3. Output filters should ignore any buckets following an EOS bucket.
  4. Output filters must process a fixed amount of data at a time, to ensure that memory consumption is not proportional to the size of the content being filtered.
  5. Output filters should be agnostic with respect to bucket types, and must be able to process buckets of unfamiliar type.
  6. After calling `ap_pass_brigade` to pass a brigade down the filter chain, output filters should call `apr_brigade_cleanup` to ensure the brigade is empty before reusing that brigade structure; output filters should never use `apr_brigade_destroy` to "destroy" brigades.
  7. Output filters must *setaside* any buckets which are preserved beyond the duration of the filter function.
  8. Output filters must not ignore the return value of `ap_pass_brigade`, and must return appropriate errors back up the filter chain.
  9. Output filters must only create a fixed number of bucket brigades for each response, rather than one per invocation.
  10. Output filters should first attempt non-blocking reads from each data bucket, and send a FLUSH bucket down the filter chain if the read blocks, before retrying with a blocking read.
-

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Developer Documentation](#)

## Apache HTTP Server 2.x Thread Safety Issues

When using any of the threaded mpms in the Apache HTTP Server 2.x it is important that every function called from Apache be thread safe. When linking in 3rd party extensions it can be difficult to determine whether the resulting server will be thread safe. Casual testing generally won't tell you this either as thread safety problems can lead to subtle race conditions that may only show up in certain conditions under heavy load.



## Global and Static Variables

When writing your module or when trying to determine if a module or 3rd party library is thread safe there are some common things to keep in mind.

First, you need to recognize that in a threaded model each individual thread has its own program counter, stack and registers. Local variables live on the stack, so those are fine. You need to watch out for any static or global variables. This doesn't mean that you are absolutely not allowed to use static or global variables. There are times when you actually want something to affect all threads, but generally you need to avoid using them if you want your code to be thread safe.

In the case where you have a global variable that needs to be global and accessed by all threads, be very careful when you update it. If, for example, it is an incrementing counter, you need to atomically increment it to avoid race conditions with other threads. You do this using a mutex (mutual exclusion). Lock the mutex, read the current value, increment it and write it back and then unlock the mutex. Any other thread that wants to modify the value has to first check the mutex and block until it is cleared.

If you are using [APR](#), have a look at the `apr_atomic_*` functions and the `apr_thread_mutex_*` functions.



This is a common global variable that holds the error number of the last error that occurred. If one thread calls a low-level function that sets `errno` and then another thread checks it, we are bleeding error numbers from one thread into another. To solve this, make sure your module or library defines `_REENTRANT` or is compiled with `-D_REENTRANT`. This will make `errno` a per-thread variable and should hopefully be transparent to the code. It does this by doing something like this:

```
#define errno (*(__errno_location()))
```

which means that accessing `errno` will call `__errno_location()` which is provided by the `libc`. Setting `_REENTRANT` also forces redefinition of some other functions to their `*_r` equivalents and sometimes changes the common `getc/putc` macros into safer function calls. Check your `libc` documentation for specifics. Instead of, or in addition to `_REENTRANT` the symbols that may affect this are `_POSIX_C_SOURCE`, `_THREAD_SAFE`, `_SVID_SOURCE`, and `_BSD_SOURCE`.



---

Not only do things have to be thread safe, but they also have to be reentrant. `strtok()` is an obvious one. You call it the first time with your delimiter which it then remembers and on each subsequent call it returns the next token. Obviously if multiple threads are calling it you will have a problem. Most systems have a reentrant version of the function called `strtok_r()` where you pass in an extra argument which contains an allocated `char *` which the function will use instead of its own static storage for maintaining the tokenizing state. If you are using [APR](#) you can use `apr_strtok()`.

`crypt()` is another function that tends to not be reentrant, so if you run across calls to that function in a library, watch out. On some systems it is reentrant though, so it is not always a problem. If your system has `crypt_r()` chances are you should be using that, or if possible simply avoid the whole mess by using `md5` instead.



The following is a list of common libraries that are used by 3rd party Apache modules. You can check to see if your module is using a potentially unsafe library by using tools such as `ldd(1)` and `nm(1)`. For [PHP](#), for example, try this:

```
% ldd libphp4.so
libsablot.so.0 => /usr/local/lib/libsablot.so.0 (0x401f6000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x402da000)
libsnmp.so.0 => /usr/lib/libsnmp.so.0 (0x402f9000)
libpdf.so.1 => /usr/local/lib/libpdf.so.1 (0x40353000)
libz.so.1 => /usr/lib/libz.so.1 (0x403e2000)
libpng.so.2 => /usr/lib/libpng.so.2 (0x403f0000)
libmysqlclient.so.11 => /usr/lib/libmysqlclient.so.11
(0x40411000)
libming.so => /usr/lib/libming.so (0x40449000)
libm.so.6 => /lib/libm.so.6 (0x40487000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x404a8000)
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x404e7000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40505000)
libssl.so.2 => /lib/libssl.so.2 (0x40532000)
libcrypto.so.2 => /lib/libcrypto.so.2 (0x40560000)
libresolv.so.2 => /lib/libresolv.so.2 (0x40624000)
libdl.so.2 => /lib/libdl.so.2 (0x40634000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40637000)
libc.so.6 => /lib/libc.so.6 (0x4064b000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
```

In addition to these libraries you will need to have a look at any libraries linked statically into the module. You can use `nm(1)` to look for individual symbols in the module.



Please drop a note to [dev@httpd.apache.org](mailto:dev@httpd.apache.org) if you have additions or corrections to this list.

Library	Version	Thread Safe?	Notes
<a href="#">ASpell/PSpell</a>		?	
<a href="#">Berkeley DB</a>	3.x, 4.x	Yes	Be careful about sharing a connection
<a href="#">bzip2</a>		Yes	Both low-level and high-level APIs. However, high-level API requires errno.
<a href="#">cdb</a>		?	
<a href="#">C-Client</a>		Perhaps	c-client uses strtok() and gethostbyname() which are not thread-safe on most C libraries. client's static data is meant to be shared. If strtok() and gethostbyname() are thread-safe on your OS, c-client <i>may</i> be thread-safe.
<a href="#">libcrypt</a>		?	
<a href="#">Expat</a>		Yes	Need a separate parser instance
<a href="#">FreeTDS</a>		?	
<a href="#">FreeType</a>		?	
<a href="#">GD 1.8.x</a>		?	
<a href="#">GD 2.0.x</a>		?	
<a href="#">gdbm</a>		No	Errors returned via a static gdbm_topen() function.
<a href="#">ImageMagick</a>	5.2.2	Yes	ImageMagick docs claim it is thread-safe since 5.2.2 (see <a href="#">Change log</a> ).
<a href="#">Imlib2</a>		?	
<a href="#">libjpeg</a>	v6b	?	
<a href="#">libmysqlclient</a>		Yes	Use mysqlclient_r library variant. For more information, please read <a href="http://dev.mysql.com/doc/mysql/6.0/en/thread-safety.html">http://dev.mysql.com/doc/mysql/6.0/en/thread-safety.html</a> .

<a href="#">Ming</a>	0.2a	?	
<a href="#">Net-SNMP</a>	5.0.x	?	
<a href="#">OpenLDAP</a>	2.1.x	Yes	Use ldap_r library variant to en
<a href="#">OpenSSL</a>	0.9.6g	Yes	Requires proper usage of CRYPTO_CRYPTO0_set_locking_callback CRYPTO0_set_id_callback
<a href="#">liboci8 (Oracle 8+)</a>	8.x,9.x	?	
<a href="#">pdflib</a>	5.0.x	Yes	PDFLib docs claim it is thread safe it has been partially thread-safe : <a href="http://www.pdflib.com/products/p">http://www.pdflib.com/products/p</a>
<a href="#">libpng</a>	1.0.x	?	
<a href="#">libpng</a>	1.2.x	?	
<a href="#">libpq (PostgreSQL)</a>	8.x	Yes	Don't share connections across t crypt ( ) calls
<a href="#">Sablotron</a>	0.95	?	
<a href="#">zlib</a>	1.1.4	Yes	Relies upon thread-safe zalloc a is to use libc's calloc/free which a

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



| | [FAQ](#) | |



HTTP SERVER PROJECT Apache HTTP Server Version 2.4

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#)

## 2.2

.

.

2.0 2.2

. 1.3

.



---

## Authn/Authz

...

...

mod\_proxy\_balancer      mod\_proxy .  
mod\_proxy\_ajp      Apache JServ Protocol  
1.3 .

mod\_filter      . , ,  
,2.0 .



---

## mod\_authnz\_ldap

```
2.0    mod_auth_ldap    2.2  Authn/Authz
      .  Require LDAP    (attribute)
      .
```

## mod\_info

```
httpd -V . ?config .
```



---

## APR 1.0 API

2.2 APR 1.0 API . APR APR-Util

. [APR](#) .

ap\_log\_cerr

IP .

httpd -t tes

## MPM

MPM ThreadStackSize .

ap\_register\_output\_filter\_protocol

ap\_filter\_protocol [mod\\_](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



HTTP SERVER PROJECT **Apache HTTP Server Version 2.4**

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.4](#) > [Programs](#)

## htdbm - Manipulate DBM password databases

htdbm is used to manipulate the DBM format files used to store usernames and password for basic authentication of HTTP users via [mod\\_authn\\_dbm](#). See the [dbmmanage](#) documentation for more information about these DBM files.



### See also

[httpd](#)

[dbmmanage](#)

[mod\\_authn\\_dbm](#)



**htdbm** [ **-TDBTYPE** ] [ **-i** ] [ **-c** ] [ **-m** | **-B** | **-d** | **-s** | **-p** ] [ **-C cost** ] [ **-t** ] [ **-v** ] *filename*  
*username*

**htdbm -b** [ **-TDBTYPE** ] [ **-c** ] [ **-m** | **-B** | **-d** | **-s** | **-p** ] [ **-C cost** ] [ **-t** ] [ **-v** ] *filename*  
*username password*

**htdbm -n** [ **-i** ] [ **-c** ] [ **-m** | **-B** | **-d** | **-s** | **-p** ] [ **-C cost** ] [ **-t** ] [ **-v** ] *username*

**htdbm -nb** [ **-c** ] [ **-m** | **-B** | **-d** | **-s** | **-p** ] [ **-C cost** ] [ **-t** ] [ **-v** ] *username password*

**htdbm -v** [ **-TDBTYPE** ] [ **-i** ] [ **-c** ] [ **-m** | **-B** | **-d** | **-s** | **-p** ] [ **-C cost** ] [ **-t** ] [ **-v** ] *filename*  
*username*

**htdbm -vb** [ **-TDBTYPE** ] [ **-c** ] [ **-m** | **-B** | **-d** | **-s** | **-p** ] [ **-C cost** ] [ **-t** ] [ **-v** ] *filename*  
*username password*

**htdbm -x** [ **-TDBTYPE** ] *filename username*

**htdbm -l** [ **-TDBTYPE** ]



## **-b**

Use batch mode; *i.e.*, get the password from the command line rather than prompting for it. This option should be used with extreme care, since **the password is clearly visible** on the command line. For script use see the `-i` option.

## **-i**

Read the password from stdin without verification (for script usage).

## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is rewritten and truncated. This option cannot be combined with the `-n` option.

## **-n**

Display the results on standard output rather than updating a database. This option changes the syntax of the command line, since the *passwdfile* argument (usually the first one) is omitted. It cannot be combined with the `-c` option.

## **-m**

Use MD5 encryption for passwords. On Windows and Netware, this is the default.

## **-B**

Use bcrypt encryption for passwords. This is currently considered to be very secure.

## **-C**

This flag is only allowed in combination with `-B` (bcrypt encryption). It sets the computing time used for the bcrypt algorithm (higher is more secure but slower, default: 5, valid: 4 to 31).

## **-d**

Use `crypt()` encryption for passwords. The default on all

platforms but Windows and Netware. Though possibly supported by `htdbm` on all platforms, it is not supported by the `httpd` server on Windows and Netware. This algorithm is **insecure** by today's standards.

**-s**

Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (ldif). This algorithm is **insecure** by today's standards.

**-p**

Use plaintext passwords. Though `htdbm` will support creation on all platforms, the `httpd` daemon will only accept plain text passwords on Windows and Netware.

**-l**

Print each of the usernames and comments from the database on stdout.

**-v**

Verify the username and password. The program will print a message indicating whether the supplied password is valid. If the password is invalid, the program exits with error code 3.

**-x**

Delete user. If the username exists in the specified DBM file, it will be deleted.

**-t**

Interpret the final parameter as a comment. When this option is specified, an additional string can be appended to the command line; this string will be stored in the "Comment" field of the database, associated with the specified username.

### ***filename***

The filename of the DBM format file. Usually without the extension `.db`, `.pag`, or `.dir`. If `-c` is given, the DBM file is

created if it does not already exist, or updated if it does exist.

***username***

The username to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.

***password***

The plaintext password to be encrypted and stored in the DBM file. Used only with the -b flag.

**-TDBTYPE**

Type of DBM file (SDBM, GDBM, DB, or "default").



One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, GNU GDBM, and Berkeley/Sleepycat DB 2/3/4. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by *filename* is the same format that `htdbm` expects to see. `htdbm` currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.



htdbm returns a zero status ("true") if the username and password have been successfully added or updated in the DBM File. htdbm returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn't match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the [Restrictions section](#)), and 7 if the file is not a valid DBM password file.



## Examples

```
htdbm /usr/local/etc/apache/.htdbm-users jsmith
```

Adds or modifies the password for user `jsmith`. The user is prompted for the password. If executed on a Windows system, the password will be encrypted using the modified Apache MD5 algorithm; otherwise, the system's `crypt ( )` routine will be used. If the file does not exist, `htdbm` will do nothing except return an error.

```
htdbm -c /home/does/public_html/.htdbm jane
```

Creates a new file and stores a record in it for user `jane`. The user is prompted for the password. If the file exists and cannot be read, or cannot be written, it is not altered and `htdbm` will display a message and return an error status.

```
htdbm -mb /usr/web/.htdbm-all jones Pwd4Steve
```

Encrypts the password from the command line (`Pwd4Steve`) using the MD5 algorithm, and stores it in the specified file.



## Security Considerations

Web password files such as those managed by `htdbm` should *not* be within the Web server's URI space -- that is, they should not be fetchable with a browser.

The use of the `-b` option is discouraged, since when it is used the unencrypted password appears on the command line.

When using the `crypt ( )` algorithm, note that only the first 8 characters of the password are used to form the password. If the supplied password is longer, the extra characters will be silently discarded.

The SHA encryption format does not use salting: for a given password, there is only one encrypted representation. The `crypt ( )` and MD5 formats permute the representation by prepending a random salt string, to make dictionary attacks against the passwords more difficult.

The SHA and `crypt ( )` formats are insecure by today's standards.



On the Windows platform, passwords encrypted with `htdbm` are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

The MD5 algorithm used by `htdbm` is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Username are limited to 255 bytes and may not include the character `:`.

---

Copyright 2017 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)