

| | [FAQ](#) | |

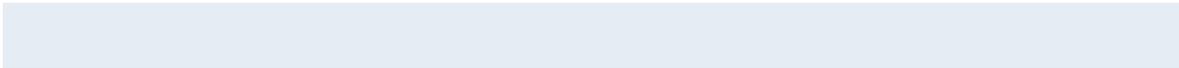


## Apache HTTP Server Version 2.2

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

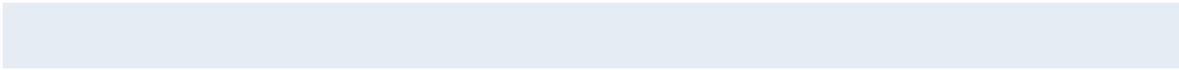
# Apache HTTP Server Version 2.2





2.0

1.3 2.0



(MPM)



(content negotiation)

(DSO)

URL

SSL/TLS

CGI Suexec

URL (rewriting)

## How-To /

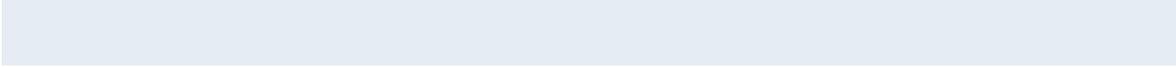
[\[redacted\]](#)

[CGI:](#)

[.htaccess](#)

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

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

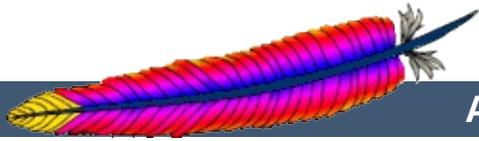


[Microsoft Windows](#)

[Novell NetWare](#)

[EBCDIC](#)

(FAQ)



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

## Upgrading to 2.2 from 2.0

In order to assist folks upgrading, we maintain a document describing information critical to existing Apache 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.

This document describes only the changes from 2.0 to 2.2. If you are upgrading from version 1.3, you should also consult the [1.3 to 2.0 upgrading document](#).

### See also

[Overview of new features in Apache 2.2](#)



## Compile-time Configuration Changes

The compilation process is very similar to the one used in version 2.0. Your old configure command line (as found in `build/config.nice` in the installed server directory) can be used in some cases. The most significant change required will be to account for changes in module names, in particular for the authentication and authorization modules. Some details of changes:

- `mod_imap` has been renamed to `mod_imagemap`
- `mod_auth` has been split up into `mod_auth_basic`, `mod_authn_file`, `mod_authz_user`, and `mod_authz_groupfile`
- `mod_access` has been renamed to `mod_authz_host`
- `mod_auth_ldap` has been renamed to `mod_authnz_ldap`
- Upgraded to require the APR 1.0 API.
- Updated bundled PCRE version to 5.0



## Runtime Configuration Changes

Your existing version 2.0 config files and startup scripts can usually be used unchanged in version 2.2. Some small adjustments may be necessary for particular configurations as discussed below. In addition, if you dynamically load the standard modules using the `LoadModule` directive, then you will need to account for the module name changes mentioned above.

If you choose to use the new default configuration file for version 2.2, you will find that it has been greatly simplified by removing all but the most essential configuration settings. A set of example configuration settings for more advanced features is present in the `conf/extra/` directory of the installed server. Default configuration files are installed in the `conf/original` directory.

Some runtime configuration changes that you may notice:

- The `apachectl` option `startssl` is no longer available. To enable SSL support, you should edit `httpd.conf` to include the relevant `mod_ssl` directives and then use `apachectl start` to start the server. An example configuration to activate `mod_ssl` has been included in `conf/extra/httpd-ssl.conf`.
- The default setting of `UseCanonicalName` is now `Off`. If you did not have this directive in your config file, you can add `UseCanonicalName On` to retain the old behavior.
- The module `mod_userdir` will no longer act on requests unless a `UserDir` directive specifying a directory name is present in the config file. To restore the old default behavior, place the directive `UserDir public_html` in your config file.
- The directive `AuthDigestFile` from `mod_auth_digest` has been merged with `AuthUserFile` and is now part of `mod_authn_file`.

- RewriteRule directives are evaluated before ProxyPass ones.



- The module [mod\\_cache](#), which was experimental in Apache 2.0, is now a standard module.
- The module [mod\\_disk\\_cache](#), which was experimental in Apache 2.0, is now a standard module.
- The module [mod\\_mem\\_cache](#), which was experimental in Apache 2.0, is now a standard module.
- The module [mod\\_charset\\_lite](#), which was experimental in Apache 2.0, is now a standard module.
- The module [mod\\_dumpio](#), which was experimental in Apache 2.0, is now a standard module.



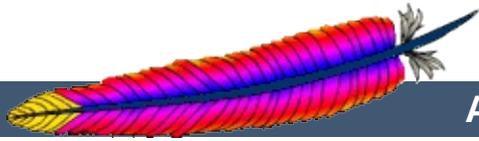
## Third-Party Modules

Many third-party modules designed for version 2.0 will work unchanged with the Apache HTTP Server version 2.2. But all modules must be recompiled before being loaded.

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

# 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**

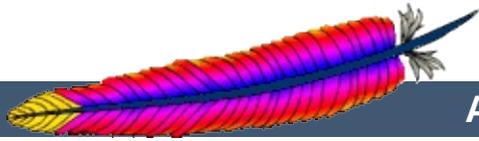
HTML ,

### **mod\_include**

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

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

[AuthDBMType](#) DBM .



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

# 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.

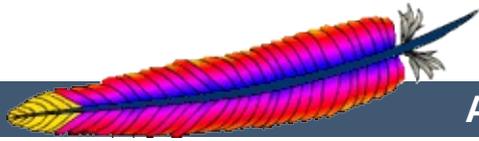
```
Copyright [yyyy] [name of copyright owner]
```

```
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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



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 .

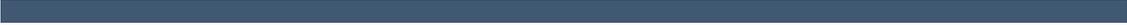
( 5.003 .) ` configure' 2.0

. 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
:

```

```

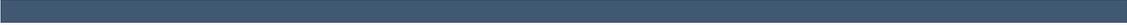
$ CC="pgcc" CFLAGS="-O2" \
./configure --prefix=/sw/pkg/apache \
--enable-rewrite=shared \
--enable-speling=shared

```

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/2.2/](#) [docs/manual/](#) .



---

:

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

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

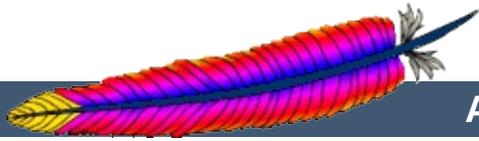
```
$ PREFIX/bin/apachectl stop
```



```
, 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](#) | |



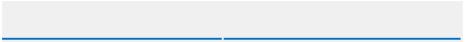
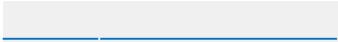
## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



Windows NT, 2000, XP ,

Windows 95 ME .



[httpd](#)

h



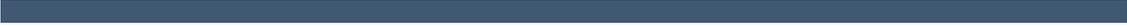
[httpd](#)

[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      ()
```



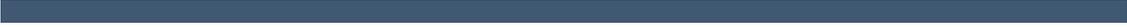


```
'  
    "    Unable to bind to Port ...".  
:
```

- root .
- .

[FAQ](#) .





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

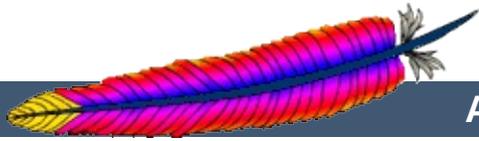


[httpd](#) [apachectl](#),

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



. NT, 2000, XP

ME



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



```
kill -TERM httpd .
kill -HUP httpd .
kill -USR1 httpd .
```

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

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

```
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 )
    root
```



---

**: 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

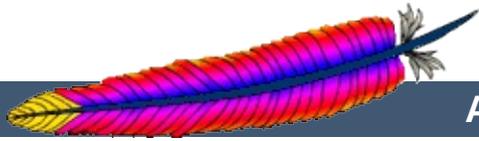
scoreboard . [ScoreBoa](#)

HTTP (KeepAlive)

. 1.2

KeepAlive

20



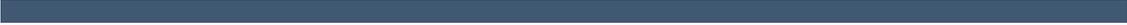
| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)





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

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



---

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



```
mod_so <IfModule>  
LoadModule
```

[base](#) .

-1 .

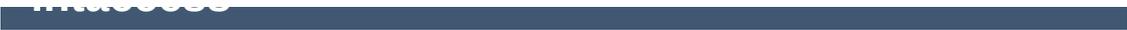


```
<Directory>
<DirectoryMatch>
<Files>
<FilesMatch>
<Location>
<LocationMatch>
<VirtualHost>
```

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

Directory, Location, Files .





```

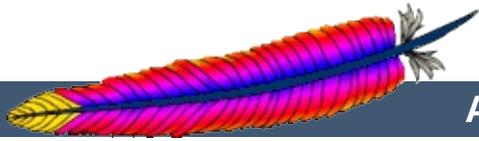
AccessFileName
AllowOverride

```

```

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

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



, , , , URL .

.htaccess .





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

```
<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>

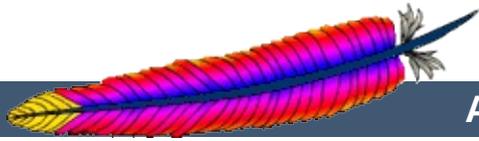
<Directory /a/b>
A
</Directory>
```

. <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](#) | |



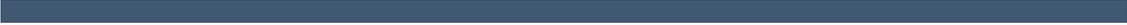
## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



core .





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

ServerAdmin ServerTokens

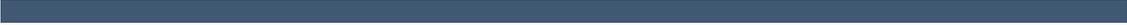
. ServerTokens HTTP .

ServerName UseCanonicalName URL .

,

.





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

. root

. (/)

.

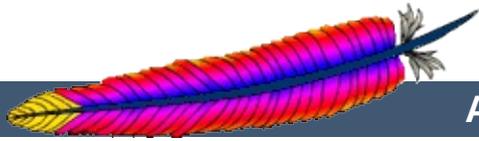


```
LimitRequestBody
LimitRequestFields
LimitRequestFieldsize
LimitRequestLine
RLimitCPU
RLimitMEM
RLimitNPROC
ThreadStackSize
```

LimitRequest\* .  
of service) .

RLimit\* . CGI  
.

ThreadStackSize Netware .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

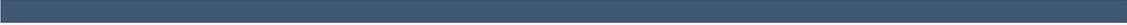


.

.

.





( 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 Yahoo
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](#)

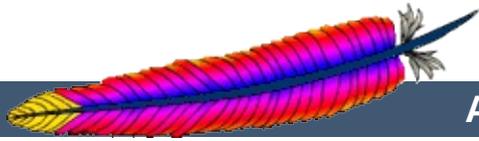
CGI .

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

[mod\\_rewrite](#)

[Rewri](#)

. [RewriteLogLevel](#) .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

**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)

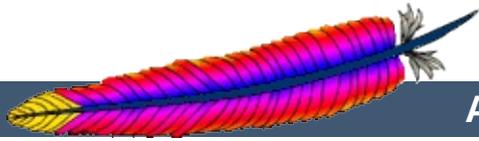
mod\_rewrite .

., mod\_rewrite

(alias), , . mod\_rewrite







| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Miscellaneous Documentation](#)



.

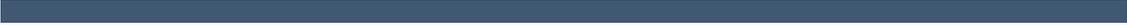
.

.

,

.





. , 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 . <
```

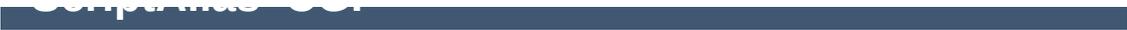


---

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](#)

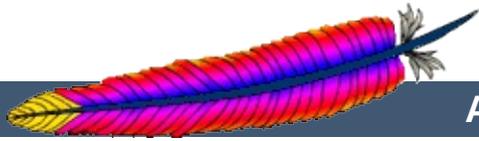
10 :

```
[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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



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. :

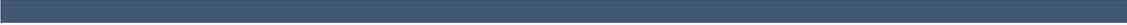
```
$ ./configure --enable-so  
$ make install
```

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

.



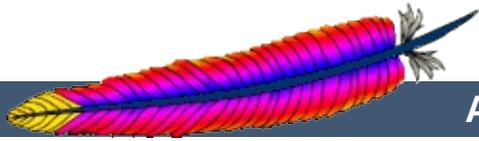
---

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

## (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)

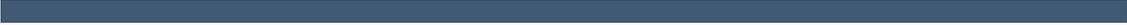
,

html

.

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





URL .

URL .

HTTP/1.1

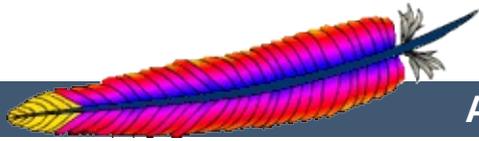
HTTP/1.0 . ,

CacheNegotiatedDocs HTTP/1.0 ( )

HTTP/1.1 .

HTTP/1.1

Vary HTTP .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



"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 . .

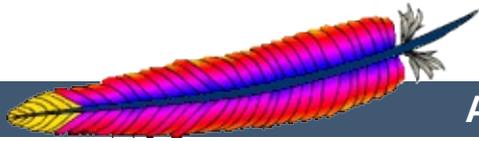
CGI REDIRECT\_ . REDIRECT\_  
REDIRECT\_HTTP\_USER\_AGENT . , HTTP\_USER\_AGE  
REDIRECT\_URL REDIRECT\_STATUS . URL URL

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

## (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

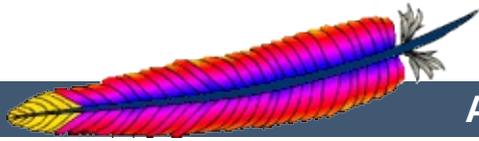
APR IPv4 ,  
:

```
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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

**(MPM)**



(Multi-Processing Module) ,

.



Apache 2.0

(Multi-Processing Modu

- mpm\_winnt Apache 1.3

POSIX

MPM

(scalability)

- worker MPM ,  
preforking MPM .  
(perchild)

MPM

MPM

MPM



---

MPMs .

MPM , MPM

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

./httpd -l MPM . MPM

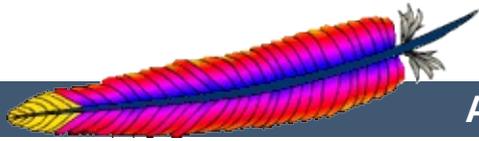
.



---

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



*(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
```

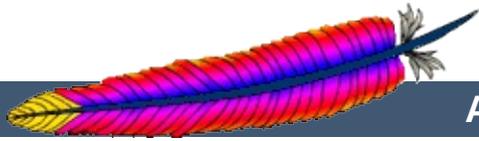
""

## /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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



<u>mod_actions</u>	<u>Action</u>
<u>mod_asis</u>	<u>AddHandler</u>
<u>mod_cgi</u>	<u>RemoveHandler</u>
<u>mod_imagemap</u>	<u>SetHandler</u>
<u>mod_info</u>	
<u>mod_mime</u>	
<u>mod_negotiation</u>	
<u>mod_status</u>	

```

                                "(handler)".
.                                , "(handled)".

```

Apache 1.1 .

```

.                                .(
___ )

```

```

, Action . :

```

- **default-handler:** def (core)
- **send-as-is:** HTTP .( mod\_asis)
- **cgi-script:** CGI .( mod\_cgi)
- **imap-file:** imagemap .( mod\_imagemap)
- **server-info:** .( mod\_info)
- **server-status:** .( mod\_status)
- **type-map:** type map .( mod\_negotiat)



---

## 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>
```



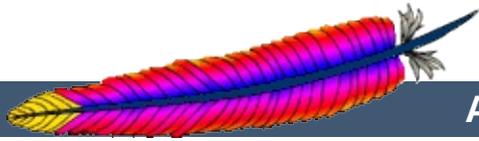
[Apache API](#) .

request

```
char *handler
```

```
, invoke_handler r->handle  
. content type .
```

```
type .
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



<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 .
```

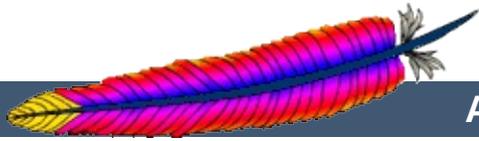
## INCLUDES

mod include Server-Side Includes

## DEFLATE

mod deflate

, mod ext filter .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

# 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



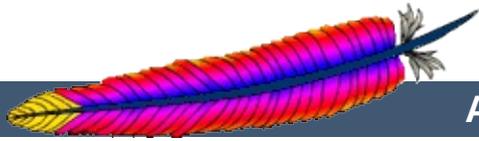
! . suEXEC ""  
wrapper .

- suEXEC
- 

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

- suEXEC PATH

- suEXEC



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Miscellaneous Documentation](#)



2.0

1.3 2.0 (scalability)

2.0





<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_SCOR
USE_SHMGET_SCOREBOARD . ( HAV
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)

```

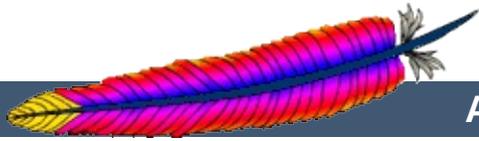
/67: accept(3, 0x001FEB74, 0x001FEB94, 1) (sleeping...)

```

```

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

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Miscellaneous Documentation](#)

## URL

*Ralf S. Engelschall* <rse@apache.org>  
1997 12

```
mod_rewrite  .  
                mod_rewrite .URL
```





```
mod_rewrite . , URL .  
URL . mod_r  
.  
: mod_rewrite ,  
.
```



---

. URL

```
:  
mod alias, mod userdir . . , [PT] .  
.htaccess .  
. .
```



## URL

```
:  
    URL . ( )  
    URL, URL . URL  
    URL .
```

```
:  
    URL HTTP .  
    /~user /u/user , /u/user
```

```
RewriteRule ^/~([^/]+)/?(.*) /u/$1/$2 [R]  
RewriteRule ^/([u|e|g|e])/([^/]+)$ /$1/$2/ [R]
```

```
:  
...  
:
```

```
RewriteCond %{HTTP_HOST} !^fully\.qualified\.domain\.name  
RewriteCond %{HTTP_HOST} !^$  
RewriteCond %{SERVER_PORT} !^80$  
RewriteRule ^/(.*) http://fully.qualified.domain.name  
RewriteCond %{HTTP_HOST} !^fully\.qualified\.domain\.name  
RewriteCond %{HTTP_HOST} !^$  
RewriteRule ^/(.*) http://fully.qualified.domain.name
```

## DocumentRoot

```
:  
    DocumentRoot URL "/" .  
    , . (
```

```
/e/www/ ( ) /e/sww/ . Docu  
/e/www/ , .
```

```
:  
URL / /e/www/ . mod  
 . (mod alias ) URL Alias .  
DocumentRoot URL  
mod_rewrite :
```

```
RewriteEngine on  
RewriteRule ^/$ /e/www/ [R]
```

```
:  
URL  
/~quux/foo/ /~quux/foo ' , foo .  
 . URL ,  
 , CGI URL .
```

```
:  
 .  
 , .  
 .  
/~quux/foo/index.html image.gif  
/~quux/image.gif !  
:
```

```
RewriteEngine on  
RewriteBase /~quux/  
RewriteRule ^foo$ foo/ [R]
```

```
.htaccess .
```

```

RewriteEngine on
RewriteBase /~quux/
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^(.+[^/])$ $1/ [R]

```

## URL

```

:
                                URL ., (
!) URL ! :
.
:
', '                                () .

```

```

user1 server_of_user1
user2 server_of_user2
:      :

```

```

map.xxx-to-host . URL
URL,

```

```

/u/user/anypath
/g/group/anypath
/e/entity/anypath

```

```

http://physical-host/u/user/anypath
http://physical-host/g/group/anypath
http://physical-host/e/entity/anypath

```

(server0 ):

```
RewriteEngine on

RewriteMap      user-to-host      txt:/path/to/map.user-to-host
RewriteMap      group-to-host     txt:/path/to/map.group-to-hos
RewriteMap      entity-to-host    txt:/path/to/map.entity-to-ho

RewriteRule     ^/u/([^/]+)/?(.*) http://${user-to-host:$1|s
RewriteRule     ^/g/([^/]+)/?(.*) http://${group-to-host:$1|s
RewriteRule     ^/e/([^/]+)/?(.*) http://${entity-to-host:$1|s

RewriteRule     ^/([uge])/([^/]+)/?$          /$1/$2/.www/
RewriteRule     ^/([uge])/([^/]+)/([^.]+.+)$ /$1/$2/.www/$3\
```

:

:

```
mod_rewrite . /~user/any{
http://newserver/~user/anypath .
```

```
RewriteEngine on
RewriteRule    ^/~(.+) http://newserver/~$1 [R,L]
```

:

```
, /~foo/anypath /home/f/foo/.www/anypath,
/~bar/anypath /home/b/bar/.www/anypath.
```

:

URL

.

RewriteEngine on

RewriteRule ^/~((**[a-z]**)[a-z0-9]+)(.\*) /home/\$2/\$1/.www\$3

:

:

RewriteRules

.: 1992

drwxrwxr-x	2	netsw	users	512	Aug	3	18:39	Audio/
drwxrwxr-x	2	netsw	users	512	Jul	9	14:37	Benchmark/
drwxrwxr-x	12	netsw	users	512	Jul	9	00:34	Crypto/
drwxrwxr-x	5	netsw	users	512	Jul	9	00:41	Database/
drwxrwxr-x	4	netsw	users	512	Jul	30	19:25	Dicts/
drwxrwxr-x	10	netsw	users	512	Jul	9	01:54	Graphic/
drwxrwxr-x	5	netsw	users	512	Jul	9	01:58	Hackers/
drwxrwxr-x	8	netsw	users	512	Jul	9	03:19	InfoSys/
drwxrwxr-x	3	netsw	users	512	Jul	9	03:21	Math/
drwxrwxr-x	3	netsw	users	512	Jul	9	03:24	Misc/
drwxrwxr-x	9	netsw	users	512	Aug	1	16:33	Network/
drwxrwxr-x	2	netsw	users	512	Jul	9	05:53	Office/
drwxrwxr-x	7	netsw	users	512	Jul	9	09:24	SoftEng/
drwxrwxr-x	7	netsw	users	512	Jul	9	12:17	System/
drwxrwxr-x	12	netsw	users	512	Aug	3	20:15	Typesetting/
drwxrwxr-x	10	netsw	users	512	Jul	9	14:08	X11/

1996 7

. "" ,

,

. ?

CGI .

:

:

CGI

/e/netsw/.www/ :

-rw-r--r--	1	netsw	users	1318	Aug	1	18:10	.wwwacl
drwxr-xr-x	18	netsw	users	512	Aug	5	15:51	DATA/
-rw-rw-rw-	1	netsw	users	372982	Aug	5	16:35	LOGFILE
-rw-r--r--	1	netsw	users	659	Aug	4	09:27	TODO
-rw-r--r--	1	netsw	users	5697	Aug	1	18:01	netsw-about
-rwxr-xr-x	1	netsw	users	579	Aug	2	10:33	netsw-acces
-rwxr-xr-x	1	netsw	users	1532	Aug	1	17:35	netsw-chang
-rwxr-xr-x	1	netsw	users	2866	Aug	5	14:49	netsw-home.
drwxr-xr-x	2	netsw	users	512	Jul	8	23:47	netsw-img/
-rwxr-xr-x	1	netsw	users	24050	Aug	5	15:49	netsw-lsdir
-rwxr-xr-x	1	netsw	users	1589	Aug	3	18:43	netsw-searc
-rwxr-xr-x	1	netsw	users	1885	Aug	1	17:41	netsw-tree.
-rw-r--r--	1	netsw	users	234	Jul	30	16:35	netsw-unlim

DATA/

net.sw

:

URL ?

, URL CGI

DocumentRoot

URL /net.sw/

/e/netsw

```

RewriteRule ^net.sw$ net.sw/ [R]
RewriteRule ^net.sw/(.*)$ e/netsw/$1

```

/e/netsw/.www/.wwwacl :

Options ExecCGI FollowSymLinks Includes MultiViews

RewriteEngine on

```

# /net.sw/
RewriteBase /net.sw/

#
# cgi
RewriteRule ^$ netsw-home.cgi [L
RewriteRule ^index\.html$ netsw-home.cgi [L

#
#
RewriteRule ^.+/(netsw-[^\]+/+.+)$ $1 [L

#
RewriteRule ^netsw-home\.cgi.* - [L
RewriteRule ^netsw-changes\.cgi.* - [L
RewriteRule ^netsw-search\.cgi.* - [L
RewriteRule ^netsw-tree\.cgi$ - [L
RewriteRule ^netsw-about\.html$ - [L
RewriteRule ^netsw-img/.*$ - [L

# cgi
#
RewriteRule !^netsw-lsdir\.cgi.* - [C
RewriteRule (.*) netsw-lsdir.cgi/$1

```

:

1. (' -') L (last)
2. ! (not) C (chain)
- 3.

**NCSA imagemap      mod\_imagemap**

```
:  
NCSA . NCSA  
mod_imagemap .  
/cgi-bin/imagemap/path/to/page.map  
. /path/to/page.map .
```

```
:  
:
```

```
RewriteEngine on  
RewriteRule ^/cgi-bin/imagemap(.*) $1 [PT]
```

```
:  
MultiViews
```

```
:  
.
```

```
RewriteEngine on  
  
# custom/ ...  
# ... !  
RewriteCond /your/docroot/dir1/{REQUEST_FILENAME}  
RewriteRule ^(.+) /your/docroot/dir1/$1 [L]  
  
# pub/ ...  
# ... !  
RewriteCond /your/docroot/dir2/{REQUEST_FILENAME}  
RewriteRule ^(.+) /your/docroot/dir2/$1 [L]  
  
# Alias ScriptAlias .
```

```
RewriteRule ^(.+) - [PT]
```

## URL

:

```
URL .  
wrapper .
```

:

```
, XSSI CGI  
/foo/S=java/bar/ /foo/bar/ STATUS  
"java" .
```

```
RewriteEngine on  
RewriteRule ^(.*)/S=([^/]+)/(.*) $1/$3 [E=STATUS:$2]
```

:

```
DNS A  
www.username.host.domain.com .
```

:

```
HTTP/1.0 , Host: HTTP HTTP/1.1  
http://www.username.host.com/  
/home/username/anypath :
```

```
RewriteEngine on  
RewriteCond %{HTTP_HOST} ^www\.[^\.]+\\.host  
RewriteRule ^(.+) %{HTTP_HOST}$1  
RewriteRule ^www\.[^\.]+\\.host\.com(.*) /home/$1$2
```



:  
URL . URL escape  
"url#anchor" URL anchor escape.  
uri\_escape() (#) escape .  
URL ?

:  
NPH-CGI . escape (NPH  
parseable headers). ( )  
URL scheme xredirect::

```
RewriteRule ^xredirect:(.+) /path/to/nph-xredirect.cgi/$1 \
[T=application/x-httpd-cgi,L]
```

xredirect: URL nph-xredirect.cgi

```
#!/path/to/perl
##
## nph-xredirect.cgi -- NPH/CGI script for extended redirec
## Copyright (c) 1997 Ralf S. Engelschall, All Rights Reser
##

$| = 1;
$url = $ENV{'PATH_INFO'};

print "HTTP/1.0 302 Moved Temporarily\n";
print "Server: $ENV{'SERVER_SOFTWARE'}\n";
print "Location: $url\n";
```

```

print "Content-type: text/html\n";
print "\n";
print "<html>\n";
print "<head>\n";
print "<title>302 Moved Temporarily (EXTENDED)</title>\n";
print "</head>\n";
print "<body>\n";
print "<h1>Moved Temporarily (EXTENDED)</h1>\n";
print "The document has moved <a HREF=\"$url\">here</a>.<p>\n";
print "</body>\n";
print "</html>\n";

##EOF##

```

```

mod_rewrite URL scheme .
, news:newsgroup

```

```

RewriteRule ^anyurl xredirect:news:newsgroup

```

```

: "" xredirect:
[R,L] .

```

## (multiplexer)

```

:
http://www.perl.com/CPAN CPAN (Comprehensive Perl
Archive Network) ? CPAN FTP
. FTP . CPAN CGI

```

```

mod_rewrite ?

```

```

:
mod_rewrite 3.0.0 " ftp:" scheme
RewriteMap .

```

```
RewriteEngine on
RewriteMap    multiplex          txt:/path/to/map.cxan
RewriteRule   ^/CxAN/(.*)       %{REMOTE_HOST}::$1
RewriteRule   ^.+\.([a-zA-Z]+)::(.*)$  ${multiplex:$1|ftp.de
```

```
##
##  map.cxan -- Multiplexing Map for CxAN
##

de      ftp://ftp.cxan.de/CxAN/
uk      ftp://ftp.cxan.uk/CxAN/
com     ftp://ftp.cxan.com/CxAN/
:
##EOF##
```

:

CGI

mod\_rewrite ?

:

TIME\_XXX .  
<STRING, >STRING, =STRING :

```
RewriteEngine on
RewriteCond   %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond   %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule   ^foo\.html$             foo.day.html
RewriteRule   ^foo\.html$             foo.night.html
```



0

:  
foo.html bar.html URL  
URL .

:  
URL URL :

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ bar.html
```

0

:  
foo.html bar.html URL  
URL URL .,

:  
URL HTTP . URL :

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ bar.html [R]
```

:  
., Netscape  
, Lynx , .

:

```

Agent" . HTTP "User-Agent" "Mozilla/3"
foo.html foo.NS.html . "Lynx"
"Mozilla" 1 2 URL foo.20.html .
foo.32.html . :

```

```

RewriteCond %{HTTP_USER_AGENT} ^Mozilla/3.*
RewriteRule ^foo\.html$ foo.NS.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx/.* [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/[12].*
RewriteRule ^foo\.html$ foo.20.html [L]

RewriteRule ^foo\.html$ foo.32.html [L]

```

```

:
mirror , HTTP .FTP
. ( )
:
Proxy Throughput ( [P])
:

```

```

RewriteEngine on
RewriteBase /~quux/
RewriteRule ^hotsheet/(.*)$ http://www.tstimpreso.com/ho

```

```

RewriteEngine on
RewriteBase /~quux/

```

```
RewriteRule ^usa-news\.html$ http://www.quux-corp.com/n
```

:

...

:

```
RewriteEngine on
RewriteCond /mirror/of/remotesite/$1 -U
RewriteRule ^http://www\.remotesite\.com/(.*)$ /mirror/of/
```

:

```
() (www2.quux-corp.com)
() (www.quux-corp.com) .
```

:

```
ALLOW Host www.quux-corp.com Port >1024 --> Host www2.quux-c
DENY Host * Port * --> Host www2.quux-c
```

proxy throughput

mod\_rewrite :

```
RewriteRule ^/~([\^/]+)/?(.*) /home/$1/.www/$2
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^/home/([\^/]+)/.www/?(.*) http://www2.quux-corp.
```

( )

:

www.foo.com      www[0-5].foo.com ( 6)  
?

:

.      DNS ,  
mod\_rewrite :

### 1. DNS Round-Robin

BIND DNS round-robin .

DNS A(address)      www[0-9].foo.com .

www0	IN	A	1.2.3.1
www1	IN	A	1.2.3.2
www2	IN	A	1.2.3.3
www3	IN	A	1.2.3.4
www4	IN	A	1.2.3.5
www5	IN	A	1.2.3.6

:

www	IN	CNAME	www0.foo.com.
	IN	CNAME	www1.foo.com.
	IN	CNAME	www2.foo.com.
	IN	CNAME	www3.foo.com.
	IN	CNAME	www4.foo.com.
	IN	CNAME	www5.foo.com.
	IN	CNAME	www6.foo.com.

, BIND .      www.foo.com ,      B]

www0-www6 .      .

DNS

www.foo.com

wwwN.foo.com

wwwN.foo.c

## 2. DNS

<http://www.stanford.edu/~schemers/docs/lbnamed/lbname>

lbnamed

DNS . DNS

Perl

5 .

## 3. Proxy Throughput Round-Robin

mod\_rewrite proxy throughput .

DNS

www0.foo.com

www.foo.com

```
www IN CNAME www0.foo.com.
```

```
www0.foo.com ., URL
5 ( www1-www5) . URL
lb.pl .
```

```
RewriteEngine on
RewriteMap lb prg:/path/to/lb.pl
RewriteRule ^/(.+)$ ${lb:$1} [P,L]
```

lb.pl :

```
#!/path/to/perl
##
## lb.pl --
##

$| = 1;

$name = "www"; #
```

```

$first = 1;          # ( 0 , 0 )
$last  = 5;          # round-robin
$domain = "foo.dom"; #

$cnt = 0;
while (<STDIN>) {
    $cnt = (($cnt+1) % ($last+1-$first));
    $server = sprintf("%s%d.%s", $name, $cnt+$first, $domain);
    print "http://$server/$_";
}

###EOF###

```

```

: ?                  www0.foo.com ?
. proxy throughput  ! SSI, CGI,
ePerl . .

```

#### 4. /TCP Round-Robin

. Cisco TCP/IP

LocalDirector

#### MIME-type,

:

```

CGI .
Action CGI          URL (          PATH_INFO
QUERY_STRINGS)    ., (secure CGI
) .scgi          cgiwrap type .
) URL              /u/user/fo
URL .              cgiwrap /~user/foo/bar.scgi/ URL
:

```

```

RewriteRule ^/[uge]/([^/]+)/\.\www/(.+)\.scgi(.*) ...

```

```
... /internal/cgi/user/cgiwrap/~$1/$2.scgi$3 [NS,T=applicat
```

```
, (URL access.log) wwwl
Glimpse) wwwidx .
. , /u/t
swwidx
```

```
/internal/cgi/user/swwidx?i=/u/user/foo/
```

```
. CGI .
.
```

:

```
CGI URL . :
```

```
RewriteRule ^/([uge])/([^/]+)(/?.*)/\^* /internal/cgi/user
RewriteRule ^/([uge])/([^/]+)(/?.*):log /internal/cgi/user
```

```
/u/user/foo/
```

```
HREF="*"
/u/user/foo/* (???)
```

```
/internal/cgi/user/wwwidx?i=/u/user/foo/
```

```
:log CGI .
```

:

```
foo.html fo
```



```
RewriteRule ^(/[uge]/[^\s]+/?.*):refresh /internal/cgi/apa
```

## URL

```
/u/foo/bar/page.html:refresh
```

## URL

```
/internal/cgi/apache/nph-refresh?f=/u/foo/bar/page.html
```

## NPH-CGI . " ;-) .

```
#!/sw/bin/perl
##
## nph-refresh -- NPH/CGI script for auto refreshing pages
## Copyright (c) 1997 Ralf S. Engelschall, All Rights Reserved
##
$| = 1;

# split the QUERY_STRING variable
@pairs = split(/&/, $ENV{'QUERY_STRING'});
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/;
    eval "\$$name = \"$value\"";
}
$QS_s = 1 if ($QS_s eq '');
$QS_n = 3600 if ($QS_n eq '');
if ($QS_f eq '') {
```

```
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: No file given\n";
    exit(0);
}
if (! -f $QS_f) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: File $QS_f not found\n";
    exit(0);
}

sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
    print "Content-type: multipart/x-mixed-replace;boundary=
    &print_http_headers_multipart_next;
}

sub print_http_headers_multipart_next {
    print "\n--$bound\n";
}

sub print_http_headers_multipart_end {
    print "\n--$bound--\n";
}

sub displayhtml {
    local($buffer) = @_ ;
    $len = length($buffer);
    print "Content-type: text/html\n";
    print "Content-length: $len\n\n";
    print $buffer;
}
```

```

sub readfile {
    local($file) = @_;
    local(*FP, $size, $buffer, $bytes);
    ($x, $x, $x, $x, $x, $x, $x, $size) = stat($file);
    $size = sprintf("%d", $size);
    open(FP, "&lt;$file");
    $bytes = sysread(FP, $buffer, $size);
    close(FP);
    return $buffer;
}

$buffer = &readfile($QS_f);
&print_http_headers_multipart_begin;
&displayhtml($buffer);

sub mystat {
    local($file) = $_[0];
    local($time);

    ($x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime) = stat($file);
    return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ($n = 0; $n &lt; $QS_n; $n++) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ($mtime ne $mtimeL) {
            $mtimeL = $mtime;
            sleep(2);
            $buffer = &readfile($QS_f);
            &print_http_headers_multipart_next;
        }
    }
}

```

```

        &displayhtml($buffer);
        sleep(5);
        $mtimeL = &mystat($QS_f);
        last;
    }
    sleep($QS_s);
}
}

&print_http_headers_multipart_end;

exit(0);

##EOF##

```

: <VirtualHost> .

: *Proxy Throughput* ( [P])

```

##
## vhost.map
##
www.vhost1.dom:80 /path/to/docroot/vhost1
www.vhost2.dom:80 /path/to/docroot/vhost2
:
www.vhostN.dom:80 /path/to/docroot/vhostN

```

```
##
```

```

## httpd.conf
##
#
#
UseCanonicalName on

#
#
# CLF
CustomLog /path/to/access_log "%{VHOST}e %h %l %u %t \"%r\
#
#
RewriteEngine on

# : URL ,
# DocumentRoot
# .
RewriteMap lowercase int:tolower
RewriteMap vhost txt:/path/to/vhost.map

#
# .
#
# 1.
RewriteCond %{REQUEST_URL} !^/commonurl1/. *
RewriteCond %{REQUEST_URL} !^/commonurl2/. *
#
RewriteCond %{REQUEST_URL} !^/commonurlN/. *
#
# 2. Host
#
# Host
RewriteCond %{HTTP_HOST} !^$
#

```

```
# 3.
RewriteCond  ${[lowercase:%{HTTP_HOST}|NONE]}  ^(.+)$
#
# 4. vhost.map
#
#      ( "NONE" )
RewriteCond  ${vhost:%1}  ^(/.*)$
#
# 5. URL
#
RewriteRule  ^(/.*)$  %1/$1  [E=VHOST:${[lowercase:%{HTTP_H
:
```



```
:
:
: Protocol " /robots.txt ? "Robot Exclusion
```

```
:
: ( ) /~c
: URL . ,
: , . User-Agent HTTP .
```

```
RewriteCond %{HTTP_USER_AGENT} ^NameOfBadRobot.*
RewriteCond %{REMOTE_ADDR} ^123\.45\.67\.[8-9]$
RewriteRule ^/~quux/foo/arc/.+ - [F]
```

```
:
: http://www.quux-corp.de/~quux/ GIF
: . , . .
```

```
:
: 100% , HTTP Referer
: .
```

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://www.quux-corp.de/~quux/
RewriteRule .*\.gif$ -
```

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !.* /foo-with-gif\.html$
RewriteRule ^inlined-in-foo\.gif$ -
```

:

?

:

>= 1.3b6:

```
RewriteEngine on
RewriteMap hosts-deny txt:/path/to/hosts.deny
RewriteCond ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND} !=NOT-F
RewriteCond ${hosts-deny:%{REMOTE_ADDR}|NOT-FOUND} !=NOT-F
RewriteRule ^/.* - [F]
```

<= 1.3b6:

```
RewriteEngine on
RewriteMap hosts-deny txt:/path/to/hosts.deny
RewriteRule ^/(.*)$ ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND}
RewriteRule !^NOT-FOUND/.* - [F]
RewriteRule ^NOT-FOUND/(.*)$ ${hosts-deny:%{REMOTE_ADDR}|N
RewriteRule !^NOT-FOUND/.* - [F]
RewriteRule ^NOT-FOUND/(.*)$ /$1
```

```
##
## hosts.deny
##
## ! .
## mod_rewrite / ,
## "-" .
##
193.102.180.41 -
bsdti1.sdm.de -
```

192.76.162.40 -

:

?

:

mod\_rewrite mod\_proxy . mod\_rewrite mod\_proxy . ...

```
RewriteCond %{REMOTE_HOST} ^badhost\.mydomain\.com$  
RewriteRule !^http://[^\./]\.mydomain.com.* - [F]
```

... user@host :

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} ^badguy@badhost\  
RewriteRule !^http://[^\./]\.mydomain.com.* - [F]
```

:

(mod\_auth\_basic Basic Auth ' ' ) .

:

:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend1@client1  
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend2@client2  
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend3@client3  
RewriteRule ^/~quux/only-for-friends/ -
```

## Referer (deflector)

:  
"Referer" HTTP URL  
?

:  
...

```
RewriteMap deflector txt:/path/to/deflector.map

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}} ^-$
RewriteRule ^.* %{HTTP_REFERER} [R,L]

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}|NOT-FOUND} !=NOT-FOU
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

... :

```
##
## deflector.map
##

http://www.badguys.com/bad/index.html -
http://www.badguys.com/bad/index2.html -
http://www.badguys.com/bad/index3.html http://somewhere.co
```

( " -" ) (URL  
URL .



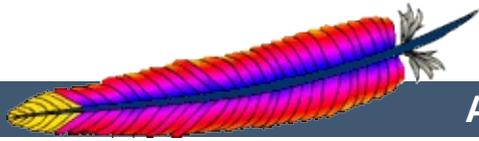
:  
FAQ: ? [mod\\_rewrite](#) ...

:  
[RewriteMap](#) . , [RewriteMap](#) .  
STDIN URL , (!) ( ) URL  
STDOUT .

```
RewriteEngine on  
RewriteMap quux-map prg:/path/to/map.quux.pl  
RewriteRule ^/~quux/(.*)$ /~quux/${quux-map:$1}
```

```
#!/path/to/perl  
  
#  
#  
$| = 1;  
  
# stdin URL  
# stdout URL  
while (<>) {  
    s|^foo/|bar/|;  
    print $_;  
}
```

/~quux/foo/... URL /~quux/bar/...



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



IP

[Redacted]

[Redacted]

[Redacted]

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 .

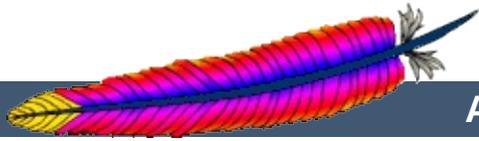
ServerPath :

:

```
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](#) | |



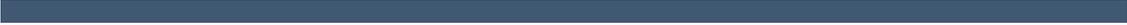
## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

**IP**







```
.  
.  
:  
• 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>

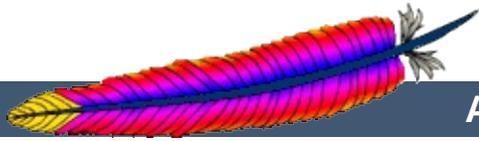
<VirtualHost www.baygroup.org>
ServerAdmin webmaster@mail.baygroup.org
DocumentRoot /groups/baygroup/www
ServerName www.baygroup.org
ErrorLog /groups/baygroup/logs/error_log
TransferLog /groups/baygroup/logs/access_log
</VirtualHost>
```

IP . ([DNS](#) )

VirtualHost  
VirtualHost

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

:  
.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



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 :
```

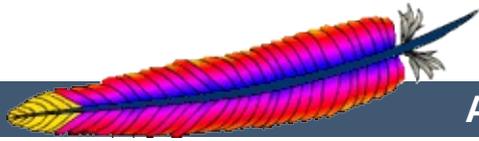
```
RewriteEngine on

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



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

Host:

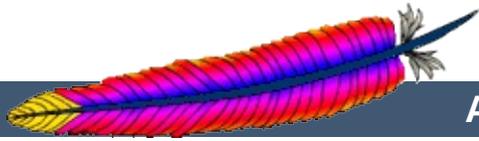
(

URL .

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



**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 .

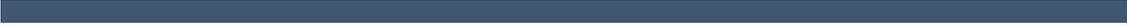
IP .

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 .

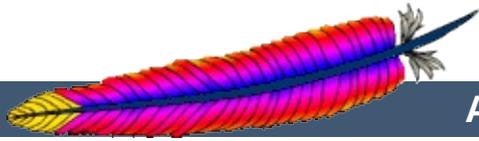


---

## DNS

:

- `VirtualHost` `.(` `.`
- `NameVirtualHost` `VirtualHost` `.`
- `ServerPath` `ServerPath` `.`  
`()` `()` `.(`  
`/abc" "ServerPath /abc/def"` `.`



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## (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

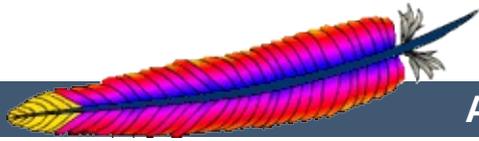
:

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

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







| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

# 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>
```



```

) .
.
abc.dom DNS
) .

```

```

<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 .

```





- 
- VirtualHost IP

- Listen IP

- ServerName

- `<VirtualHost _default_*>`



DNS . 1.2  
. IP

DNS

IP DNS

DNS . (FTP  
)

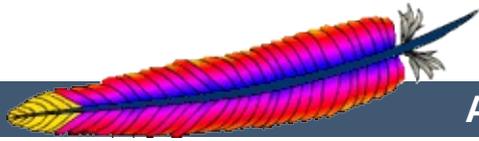
TCP wrapper "-" DNS

IP DNS

HTTP/1.1

Host IP

DNS . 1997 3



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: An Introduction

*The nice thing about standards is that there are so many to choose from. And if you really don't like all the standards you just have to wait another year until the one arises you are looking for.*

-- A. Tanenbaum, "Introduction to Computer Networks"

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.

The presented content is mainly derived, with the author's permission, from the article [Introducing SSL and Certificates using SSLey](#) by [Frederick J. Hirsch](#), of The Open Group Research Institute, which was published in [Web Security: A Matter of Trust](#), World Wide Web Journal, Volume 2, Issue 3, Summer 1997. Please send any positive feedback to [Frederick Hirsch](#) (the original article author) and all negative feedback to [Ralf S. Engelschall](#) (the `mod_ssl` author).



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**

<b>DN Field</b>	<b>Abbrev.</b>	<b>Description</b>	<b>Example</b>
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 [X208] [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
FTATBgNVBAGTDFNuYWt1IERlc2VydDETMDEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChM0U25ha2UgT21sLCBmdGQxHjAcBgNVBAsTFUN1cnRpZm1jYXR1IEF1dGhv
cm10eTEVMBMGA1UEAxMMU25ha2UgT21sIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTgxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAkGA1UEBhMCWFkxFTATBgNVBAGTDFNuYWt1IERlc2VydDETMDEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChM0U25ha2UgT21sLCBmdGQxHjAcBgNVBAsTD1d1YnN1
cnZlcjBUZWFtMRkwFwYDVQQDExB3d3cuc25ha2VvaWwuZG9tMR8wHQYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zch+da+rPTx/DPRp3xGjHZ4GG6pCmvADIETBtKBFACZ64n+Dy7Np8b
vKR+yy5DGQiijsH1D/j8H1GE+q4TZ80Fk7BNBFazHxFbYI40KMicXdKzdif1yfaa
lWoANf1Az1SdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWGCSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAA0B
gQAZUIHAL4D09oE6Lv2k56Gp380BDuILvwLg1v1KL8mQR+KFjghCrtpqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdKpfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHZICxBVC1lnHyYgJDuAMhe396lYAn8bClD1/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	Browser Support
SSL v2.0	Vendor Standard (from Netscape Corp.) <a href="#">[SSL2]</a>	First SSL protocol for which implementations exist	- NS Navigator 1.x/2.x - MS IE 3.x - Lynx/2.8+OpenSSL
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	- NS Navigator 2.x/3.x/4.x - MS IE 3.x/4.x - Lynx/2.8+OpenSSL
TLS v1.0	Proposed Internet Standard	Revision of SSL 3.0 to update the MAC layer to HMAC, add block	- Lynx/2.8+OpenSSL

	(from IETF) <a href="#">[TLS1]</a>	padding for block ciphers, message order standardization and more alert messages.	
--	--	--	--

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).

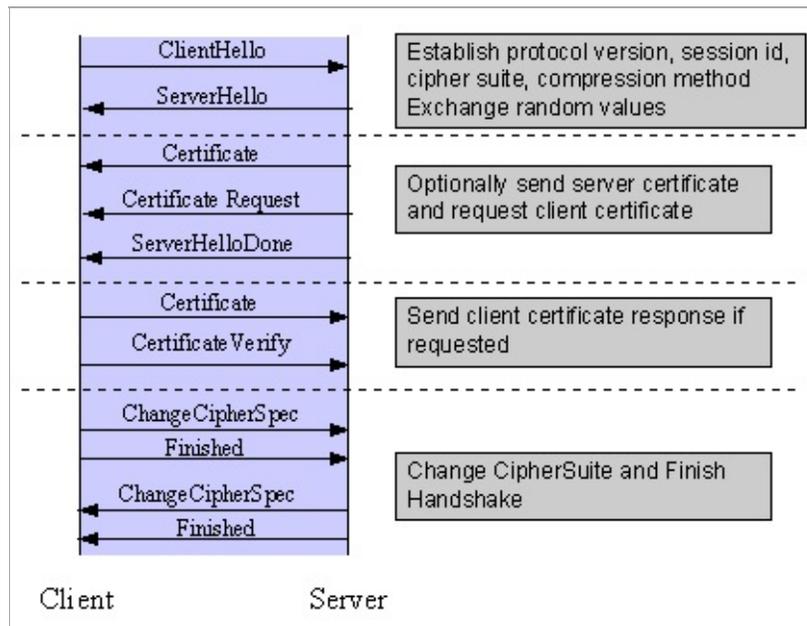
## Establishing a Session

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.

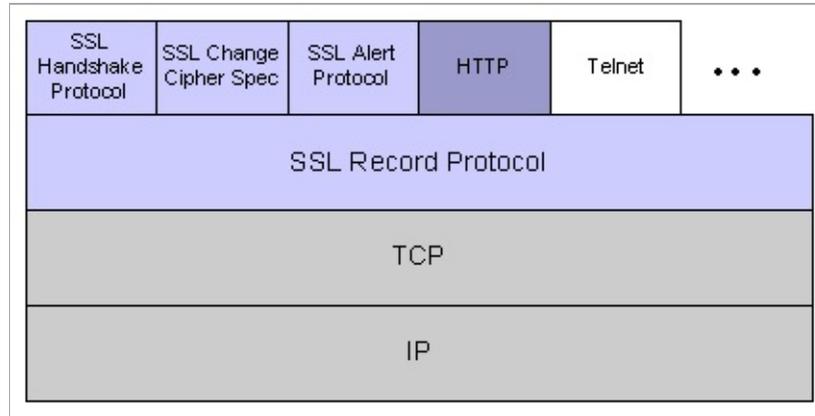
## Handshake Sequence Protocol

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.

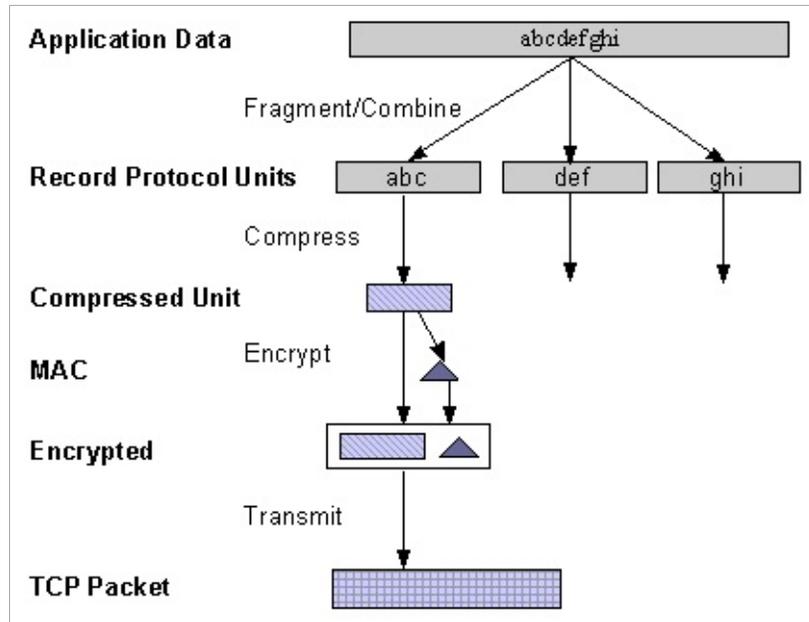


**Figure 2:** *SSL Protocol Stack*

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.

## [X208]

ITU-T Recommendation X.208, *“Specification of Abstract Syntax Notation One (ASN.1)”*, 1988. See for instance <http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.208-198811-I>.

## [X509]

ITU-T Recommendation X.509, *“The Directory - Authentication Framework”*. See for instance <http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-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://ietf.org/rfc/rfc2045.txt>.

## [SSL2]

Kipp E.B. Hickman, *“The SSL Protocol”*, 1995. See [http://www.netscape.com/eng/security/SSL\\_2.html](http://www.netscape.com/eng/security/SSL_2.html).

## [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>.

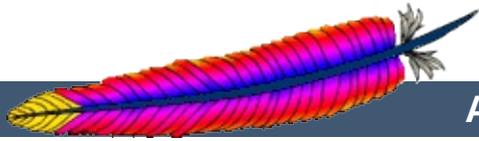
## [TLS1]

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

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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](#).

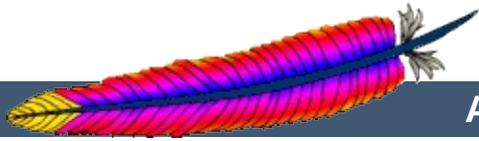
**Table 3: Custom Log Cryptography Function**

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: How-To

*The solution to this problem is trivial and is left as an exercise for the reader.*

-- Standard textbook cookie

How to solve particular security problems for an SSL-aware webserver is not always obvious because of the interactions between SSL, HTTP and Apache's way of processing requests. This chapter gives instructions on how to solve some typical situations. Treat it as a first step to find out the final solution, but always try to understand the stuff before you use it. Nothing is worse than using a security solution without knowing its restrictions and how it interacts with other systems.



- [How can I create a real SSLv2-only server?](#)
- [How can I create an SSL server which accepts strong encryption only?](#)
- [How can I create an SSL server which accepts strong encryption only, but allows export browsers to upgrade to stronger encryption?](#)
- [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 a real SSLv2-only server?

The following creates an SSL server which speaks only the SSLv2 protocol and its ciphers.

### httpd.conf

```
SSLProtocol -all +SSLv2
SSLCipherSuite SSLv2:+HIGH:+MEDIUM:+LOW:+EXP
```

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

The following enables only the seven strongest ciphers:

### httpd.conf

```
SSLProtocol all
SSLCipherSuite HIGH:MEDIUM
```

## How can I create an SSL server which accepts strong encryption only, but allows export browsers to upgrade to stronger encryption?

This facility is called Server Gated Cryptography (SGC) and requires a Global ID server certificate, signed by a special CA

certificate from Verisign. This enables strong encryption in 'export' versions of browsers, which traditionally could not support it (because of US export restrictions).

When a browser connects with an export cipher, the server sends its Global ID certificate. The browser verifies this, and can then upgrade its cipher suite before any HTTP communication takes place. The problem lies in allowing browsers to upgrade in this fashion, but still requiring strong encryption. In other words, we want browsers to either start a connection with strong encryption, or to start with export ciphers but upgrade to strong encryption before beginning HTTP communication.

This can be done as follows:

### **httpd.conf**

```
# allow all ciphers for the initial handshake,  
# so export browsers can upgrade via SGC facility  
SSLCipherSuite  
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL  
  
<Directory /usr/local/apache2/htdocs>  
# but finally deny all browsers which haven't upgraded  
SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128  
</Directory>
```

## **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:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

<Location /strong/area>
# but https://hostname/strong/area/ and below
# requires strong ciphers
SSLCipherSuite HIGH:MEDIUM
</Location>
```



- [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.

### httpd.conf

```
# require a client certificate which has to be directly
# 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](#):

## httpd.conf

```
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:

## httpd.conf

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

<Directory /usr/local/apache2/htdocs/secure/area>
SSLVerifyClient require
SSLVerifyDepth 5
SSLOptions +FakeBasicAuth
SSLRequireSSL
AuthName "Snake Oil Authentication"
AuthType Basic
AuthBasicProvider file
AuthUserFile /usr/local/apache2/conf/httpd.passwd
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:

### httpd.conf

```
SSLVerifyClient      none  
SSLCACertificateFile conf/ssl.crt/ca.crt  
SSLCACertificatePath conf/ssl.crt  
  
<Directory /usr/local/apache2/htdocs/secure/area>  
  SSLVerifyClient    require  
  SSLVerifyDepth     5  
  SSLOptions          +FakeBasicAuth  
  SSLRequireSSL  
  SSLRequire          %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \  
                    and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"  
</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.

## httpd.conf

```
SSLCertificateFile conf/ssl.crt/company-ca.crt

<Directory /usr/local/apache2/htdocs>
#   Outside the subarea only Intranet access is granted
Order                deny,allow
Deny                 from all
Allow                from 192.168.1.0/24
</Directory>

<Directory /usr/local/apache2/htdocs/subarea>
#   Inside the subarea any Intranet access is allowed
#   but from the Internet only HTTPS + Strong-Cipher + Password
#   or the alternative HTTPS + Strong-Cipher + Client-Certificate

#   If HTTPS is used, make sure a strong cipher is used.
#   Additionally allow client certs as alternative to basic auth.
SSLVerifyClient     optional
SSLVerifyDepth      1
SSLOptions           +FakeBasicAuth +StrictRequire
SSLRequire           %{SSL_CIPHER_USEKEYSIZE} >= 128

#   Force clients from the Internet to use HTTPS
RewriteEngine       on
RewriteCond         %{REMOTE_ADDR} !^192\.168\.1\.[0-9]+$
RewriteCond         %{HTTPS} !=on
RewriteRule         .* - [F]

#   Allow Network Access and/or Basic Auth
Satisfy             any

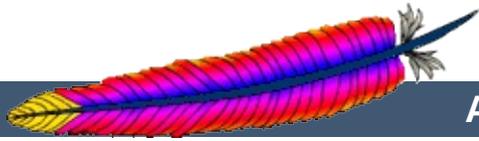
#   Network Access Control
Order                deny,allow
Deny                 from all
Allow                192.168.1.0/24

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

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [SSL/TLS](#)

## SSL/TLS Strong Encryption: FAQ

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

-- Claude Levi-Strauss

This chapter is a collection of frequently asked questions (FAQ) and corresponding answers following the popular USENET tradition. Most of these questions occurred on the Newsgroup [comp.infosystems.www.servers.unix](mailto:comp.infosystems.www.servers.unix) or the mod\_ssl Support Mailing List [modssl-users@modssl.org](mailto:modssl-users@modssl.org). They are collected at this place to avoid answering the same questions over and over.

Please read this chapter at least once when installing mod\_ssl or at least search for your problem here before submitting a problem report to the author.



- [What is the history of mod\\_ssl?](#)
- [mod\\_ssl and Wassenaar Arrangement?](#)

## What is the history of mod\_ssl?

The mod\_ssl v1 package was initially created in April 1998 by [Ralf S. Engelschall](#) via porting [Ben Laurie's Apache-SSL 1.17](#) source patches for Apache 1.2.6 to Apache 1.3b6. Because of conflicts with Ben Laurie's development cycle it then was re-assembled from scratch for Apache 1.3.0 by merging the old mod\_ssl 1.x with the newer Apache-SSL 1.18. From this point on mod\_ssl lived its own life as mod\_ssl v2. The first publicly released version was mod\_ssl 2.0.0 from August 10th, 1998.

After US export restrictions on cryptographic software were loosened, [mod\\_ssl](#) became part of the Apache HTTP Server with the release of Apache httpd 2.

## Is mod\_ssl affected by the Wassenaar Arrangement?

First, let us explain what *Wassenaar and its Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies* is: This is a international regime, established in 1995, to control trade in conventional arms and dual-use goods and technology. It replaced the previous *CoCom* regime. Further details on both the Arrangement and its signatories are available at <http://www.wassenaar.org/>.

In short, the aim of the Wassenaar Arrangement is to prevent the build up of military capabilities that threaten regional and international security and stability. The Wassenaar Arrangement controls the export of cryptography as a dual-use good, that is, something that has both military and civilian applications. However, the Wassenaar Arrangement also provides an

exemption from export controls for mass-market software and free software.

In the current Wassenaar *List of Dual Use Goods and Technologies And Munitions*, under “GENERAL SOFTWARE NOTE (GSN)” it says “The Lists do not control "software" which is either: 1. [...] 2. "in the public domain".” And under “DEFINITIONS OF TERMS USED IN THESE LISTS” we find “In the public domain” defined as “"technology" or "software" which has been made available without restrictions upon its further dissemination. Note: Copyright restrictions do not remove "technology" or "software" from being "in the public domain".”

So, both mod\_ssl and OpenSSL are “in the public domain” for the purposes of the Wassenaar Arrangement and its “*List of Dual Use Goods and Technologies And Munitions List*”, and thus not affected by its provisions.



- [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}/$1 [R,L]
RewriteRule ^/(.*)_NOSSL$ http://%{SERVER_NAME}/$1 [R,L]
```

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?](#)
- [Why do connections fail with an "alert bad certificate" error?](#)
- [Why does my 2048-bit private key not work?](#)
- [Why is client authentication broken after upgrading from SSLeay version 0.8 to 0.9?](#)
- [How can I convert a certificate from PEM to DER format?](#)
- [Why can't I find the getca or getverisign programs mentioned by Verisign, for installing my Verisign certificate?](#)
- [Can I use the Server Gated Cryptography \(SGC\) facility \(aka Verisign Global ID\) with mod\\_ssl?](#)
- [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/th  
SSLCertificateKeyFile /path/to/th
```

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 1024
```

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 more information about commercial CAs see the following locations:

1. Verisign

<http://digitalid.verisign.com/server/apacheNotice.htm>

2. Thawte

<http://www.thawte.com/>

3. CertiSign Certificadora Digital Ltda.  
<http://www.certsign.com.br>
4. IKS GmbH  
<http://www.iks-jena.de/leistungen/ca/>
5. Uptime Commerce Ltd.  
<http://www.uptimecommerce.com>
6. BelSign NV/SA  
<http://www.belsign.be>

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 1024
```

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
```

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 pass-

phrase. 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
```

## **Why do connections fail with an "alert bad certificate" error?**

Errors such as `OpenSSL: error:14094412: SSL routines:SSL3_READ_BYTES:sslv3 alert bad certificate` in the SSL logfile, are usually caused by a browser which is unable to handle the server certificate/private-key. For example, Netscape Navigator 3.x is unable to handle RSA key lengths not equal to 1024 bits.

## **Why does my 2048-bit private key not work?**

The private key sizes for SSL must be either 512 or 1024 bits, for compatibility with certain web browsers. A keysize of 1024 bits is recommended because keys larger than 1024 bits are incompatible with some versions of Netscape Navigator and Microsoft Internet Explorer, and with other browsers that use RSA's BSAFE cryptography toolkit.

## **Why is client authentication broken after upgrading from SSLeay version 0.8 to 0.9?**

The CA certificates under the path you configured with `SSLCACertificatePath` are found by SSLeay through hash symlinks. These hash values are generated by the ``openssl`

x509 -noout -hash' command. However, the algorithm used to calculate the hash for a certificate changed between SSLeay 0.8 and 0.9. You will need to remove all old hash symlinks and create new ones after upgrading. Use the Makefile provided by [mod\\_ssl](#).

## **How can I convert a certificate from PEM to DER format?**

The default certificate format for SSLeay/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 can't I find the `getca` or `getverisign` programs mentioned by Verisign, for installing my Verisign certificate?**

Verisign has never provided specific instructions for Apache+`mod_ssl`. The instructions provided are for C2Net's Stronghold (a commercial Apache based server with SSL support).

To install your certificate, all you need to do is to save the certificate to a file, and give the name of that file to the [SSLCertificateFile](#) directive. You will also need to give it the key file. For more information, see the [SSLCertificateKeyFile](#) directive.

## **Can I use the Server Gated Cryptography (SGC) facility (aka Verisign Global ID) with `mod_ssl`?**

Yes. [mod\\_ssl](#) has included support for the SGC facility since version 2.1. No special configuration is required - just use the Global ID as your server certificate. The *step up* of the clients is then automatically handled by [mod\\_ssl](#) at run-time.

## **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 \(MSIE\)?](#)
- [Why do I get I/O errors, or the message "Netscape has encountered bad data from the server", when connecting via HTTPS to an Apache+mod\\_ssl server with Netscape Navigator?](#)
- [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 MD5
2. RC4 with MD5 (export version restricted to 40-bit key)
3. RC2 with MD5
4. RC2 with MD5 (export version restricted to 40-bit key)

5. IDEA with MD5
6. DES with MD5
7. Triple-DES with MD5

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 `httpd.conf-dist`) 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.

## **Why is it not 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 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.*" \
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.

The next problem is that 56bit export versions of MSIE 5.x browsers have a broken SSLv3 implementation, which interacts badly with OpenSSL versions greater than 0.9.4. You can accept this and require your clients to upgrade their browsers, you can downgrade to OpenSSL 0.9.4 (not advised), or you can work around this, accepting that your workaround will affect other browsers too:

```
SSLProtocol all -SSLv3
```

will completely disables the SSLv3 protocol and allow those browsers to work. A better workaround is to disable only those ciphers which cause trouble.

```
SSLCipherSuite
```

ALL: !ADH: !EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP

This also allows the broken MSIE versions to work, but only removes the newer 56bit TLS ciphers.

Another problem with MSIE 5.x clients is that they refuse to connect to URLs of the form `https://12.34.56.78/` (where IP-addresses are used instead of the hostname), if the server is using the Server Gated Cryptography (SGC) facility. This can only be avoided by using the fully qualified domain name (FQDN) of the website in hyperlinks instead, because MSIE 5.x has an error in the way it handles the SGC negotiation.

And finally there are versions of MSIE which seem to require that an SSL session can be reused (a totally non standard-conforming behaviour, of course). Connecting with those MSIE versions only work if a SSL session cache is used. So, as a work-around, make sure you are using a session cache (see the [SSLSessionCache](#) directive).

### **Why do I get I/O errors, or the message "Netscape has encountered bad data from the server", when connecting via HTTPS to an Apache+mod\_ssl server with Netscape Navigator?**

This usually occurs when you have created a new server certificate for a given domain, but had previously told your browser to always accept the old server certificate. Once you clear the entry for the old certificate from your browser, everything should be fine. Netscape's SSL implementation is correct, so when you encounter I/O errors with Netscape Navigator it is usually caused by the configured certificates.

### **Why do I get handshake failures with Java-based clients when using a certificate with more than 1024**

## bits?

Beginning with version 2.2.30, [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/2.2/ssl/ssl\\_faq.html](http://httpd.apache.org/docs/2.2/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.

### Postings from the modssl-users Support Mailing List

<http://www.modssl.org/support/>

Search for your problem in the archives of the modssl-users mailing list. You're probably not the first person to have had this problem!

## 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 modssl-users Support Mailing*

### *List*

[modssl-users@modssl.org](mailto:modssl-users@modssl.org)

This is the preferred way of submitting your problem report, because this way, others can see the problem, and learn from any answers. You must subscribe to the list first, but you can then easily discuss your problem with both the author and the whole mod\_ssl user community.

### 2. *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.

### 3. *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 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+mod\_ssl+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+mod\_ssl+OpenSSL 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.

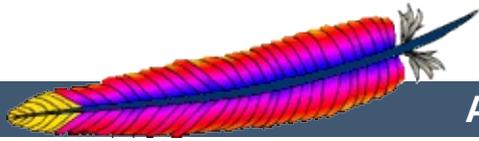
## **How do I get a backtrace, to help find the reason for my core dump?**

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.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [How-To / Tutorials](#)

# (Authentication), (Authorization), (Access Control)



(authentication)

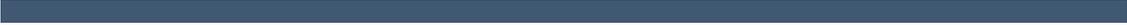
. (authorization)

.



<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

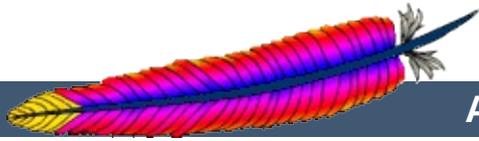
```
Order deny,allow  
Deny from all  
Allow from dev.example.com
```

## Allow ,



## mod\_auth\_basic mod\_authz\_host

.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

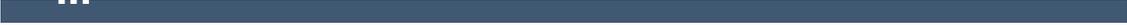
CGI::Lite

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

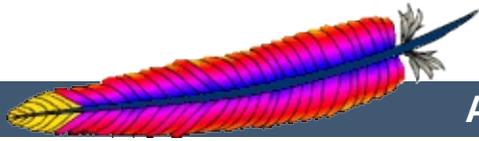
CGI .

[Common Gateway Interface RFC](#) .

CGI

, , CGI

CGI



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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" -->
```

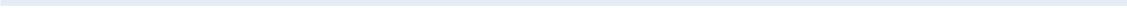
element timefmt .

**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}" -->
```

if, elif, else, endif .

SSI .

```
<!--#if expr="test_condition" -->
<!--#elif expr="test_condition" -->
<!--#else -->
<!--#endif -->
```

*test\_condition* . , ``"  
.( .) , [mod\\_j](#)  
.  
.

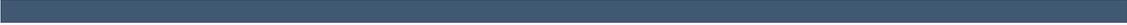
```
BrowserMatchNoCase macintosh Mac
BrowserMatchNoCase MSIE InternetExplorer
```

Internet Explorer ``Mac" ``InternetE  
.  
SSI .

```
<!--#if expr="{Mac} && {InternetExplorer}" -->
<!--#else -->
  JavaScript
<!--#endif -->
```

IE . JavaScript  
IE .  
( )  
CGI .



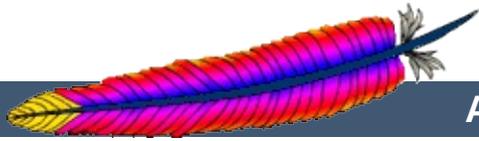


SSI CGI

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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

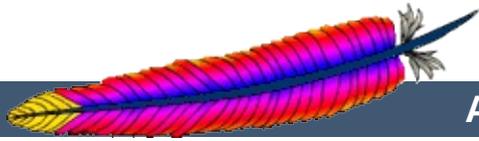
AllowOverride

AllowOverride None

None

.htaccess

.ht



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 .

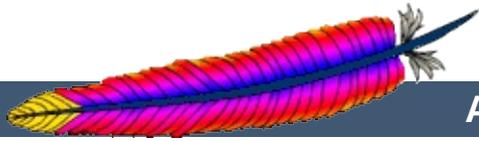
```
http://example.com/~rbowen/cgi-bin/example.cgi
```



,  
[AllowOverride](#)

[.htaccess](#)

.htaccess .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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

`LoadModule`

Windows :

- Windows , , 2.  
:

MaxRequestsPerChild:

MaxRequestsPerChild 0

```
LoadModule httpd_module modules/mod_httpd.so
```

ThreadsPerChild:

ThreadsPerChild 50.

- Windows .

- Windows  
\\Apache2\modules  
LoadModule . , status  
status ) :

```
LoadModule status_module modules/mod_status.so
```

`LoadModule`

- Microsoft IIS Windows Application Programming Interface) (, )  
 . [ISAPI](#) . 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 . (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 :
```

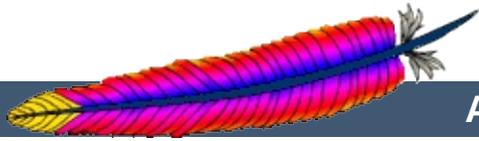
```
http://localhost/
```

```
error.log . DNS (Domain Name Service)
URL :
```

```
http://127.0.0.1/
```

```
conf . , Windows NT service
```

```
TCP/IP
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 .

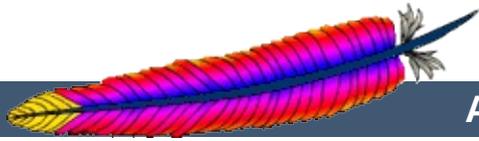
:

- *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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Platform Specific Notes](#)

# Novell NetWare

Novell NetWare 6.0 2.0 ,

dev-httpd

[\(FAQ\)](#) ,

, NetWare

[novell.devsup.webserver](http://novell.devsup.webserver)

( )

[NetWare](#)



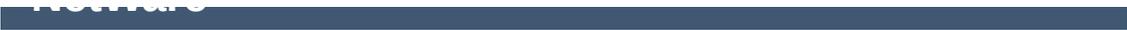
---

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
```

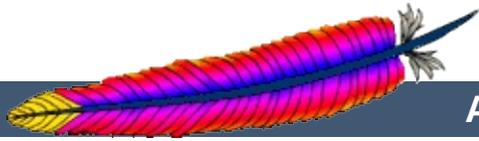
- test\_char.h chartables.c \httpd-2.0\os\netware .
- \httpd-2.0 "gmake -f nwgnumakefile"

```
gmake -f nwgnumakefile install
```

```
install .
```

## make

- gmake -f nwgnumakefile \release .
- gmake -f nwgnumakefile DEBUG=1 \debug .
- gmake -f nwgnumakefile install \dist\Apache2 , , .
- gmake -f nwgnumakefile installdev install, \lib \include import .
- gmake -f nwgnumakefile clean DEBUG \release \debug .
- gmake -f nwgnumakefile clobber\_all clean .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 "" .

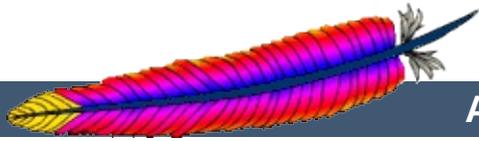
<ftp://ftp.cup.hp.com/dist/networking/misc/listenq>

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Platform Specific Notes](#)

# EBCDIC

. .

2.0 . ,



### 1.3 EBCDIC

(-ASCII)

([BS2000/OSD](#) [SIEMENS](#) .  
POSIX ).

S

- [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 .

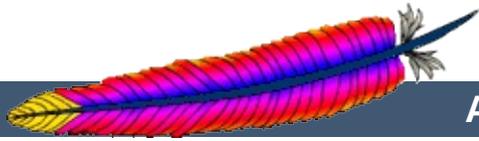


<u>core</u>	+	
<u>mod_access</u>	+	
<u>mod_actions</u>	+	
<u>mod_alias</u>	+	
<u>mod_asis</u>	+	
<u>mod_auth</u>	+	
<u>mod_auth_anon</u>	+	
<u>mod_auth_dbm</u>	?	libdb.a
<u>mod_autoindex</u>	+	
<u>mod_cern_meta</u>	?	
<u>mod_cgi</u>	+	
mod_digest	+	
<u>mod_dir</u>	+	
<u>mod_so</u>	-	
<u>mod_env</u>	+	
<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="#">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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 .

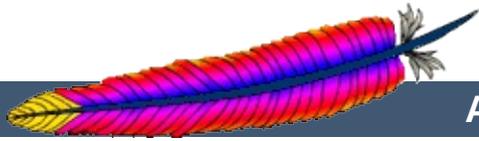
**-X**  
 . , .

[Windows](#) :

**-k install|config|uninstall**  
 Windows NT ; ; .

**-n *name***  
*name*.

**-w**  
 .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 > .

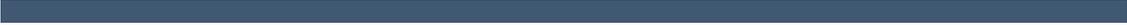
**-X *proxy[:port]***

.

**-y** *<tr>-attributes*  
    <tr> .

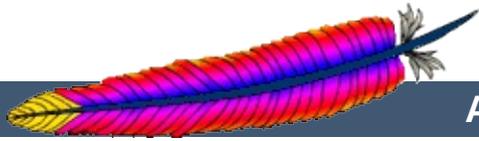
**-z** *<td>-attributes*  
    <td> .





. ' '

.  
HTTP/1.x ; " .  
; , ab



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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**

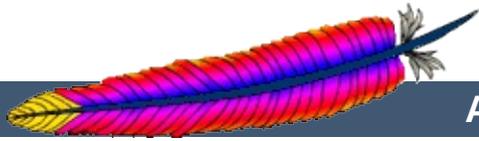
apachectl -k start -DSSL .

SSL            httpd.conf   <IfDefine> .

---

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

|| [FAQ](#) ||



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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
```

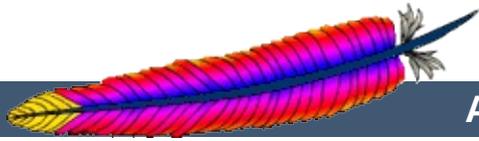
```
apxs
```

```
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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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_disk_cache 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_disk_cache .
--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* .

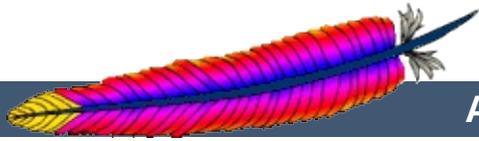
*includedir*

**LDFLAGS**

. ,

*libdir*

*-Llibc*



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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**

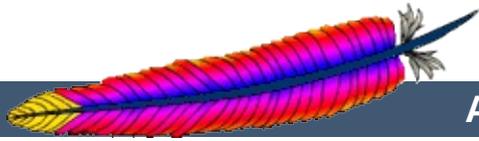
*adduser , filename username .*

**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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Programs](#)

## htcacheclean -

```
htcacheclean mod disk cache .  
(daemon) .  
. TERM INT .
```

mod disk cache



---

```
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**

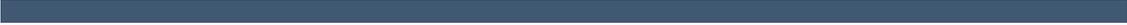
*limit* .  
K, M .

( B )

**-i**

-d .





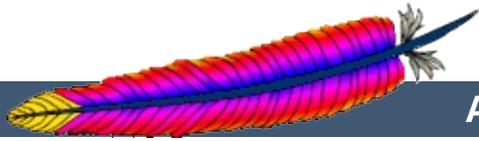
htcacheclean

(") 0 ,

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Programs](#)

# htdigest - digest authentication

htdigest HTTP digest authentication , ,  
htdigest .

manpage . [httpd](#) digest authentication  
<http://httpd.apache.org/> .

[httpd](#)  
mod\_auth\_digest



---

```
htdigest [ -c ] passwdfile realm username
```



---

**-C**

*passwdfile* . *passwdfile* .

***passwdfile***

, , . -C ,

.

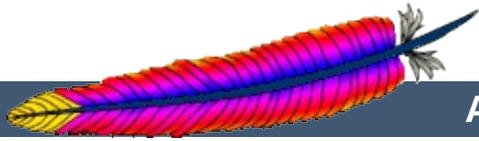
***realm***

.

***username***

*passwdfile* .

*username* .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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/doe/public_html/.htpasswd jane
```

```
jane . .  
htpasswd .
```

```
htpasswd -mb /usr/web/.htpasswd-all jones Pwd4Steve
```

```
( Pwd4Steve) MD5 .
```



---

htpasswd

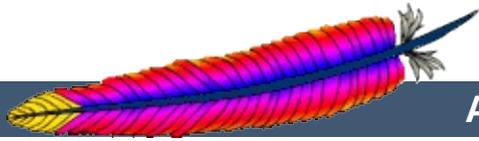
URI

.,

-b .







| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Programs](#)

# logresolve - IP-

logresolve IP- .

., IP .

.

IP, .



---

```
logresolve [ -s filename ] [ -c ] < access_log >  
access_log.new
```



---

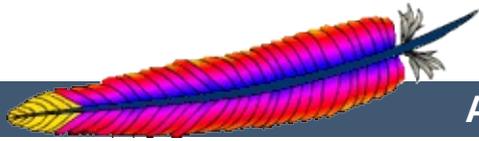
**-s** *filename*

.

**-c**

logresolve DNS :IP

IP .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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

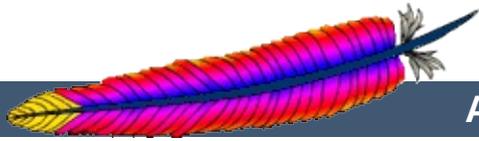
.



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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Programs](#)

## Other Programs



manpage ,



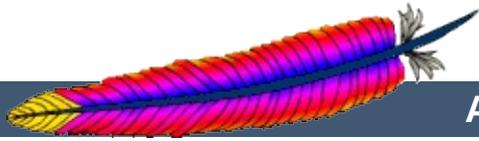
log\_server\_static

perl cron .



## Optimizing the log output

```
perl -e 'print "log\n";' + " .log" .
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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)  
( ) .

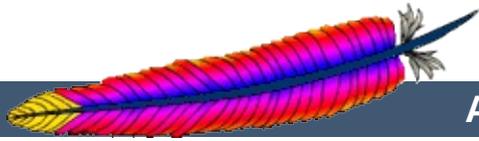
**ISO 3166-1**

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](#) | |



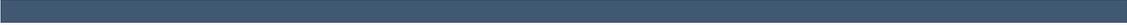
## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



.





.



---

**MPM**

"MPM"

MP

**Base**

"Base"

**Extension**

"Extension"

**Experimental**

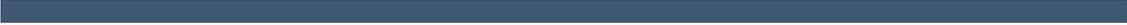
"Experimental"

**External**

"External"

(" ").

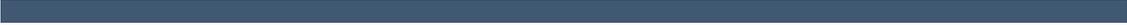




.

<IfModule> .

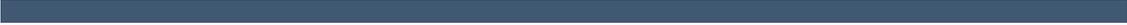




, [LoadModule](#) . |

.

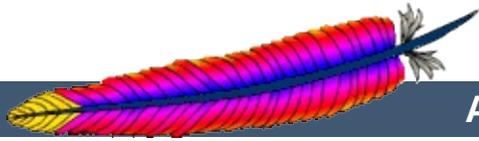




2 ,

• ,

.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



(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***

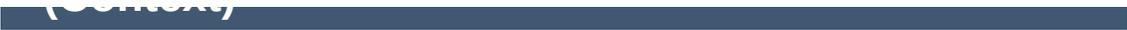


(~~Content~~)

( , .)  
"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> .

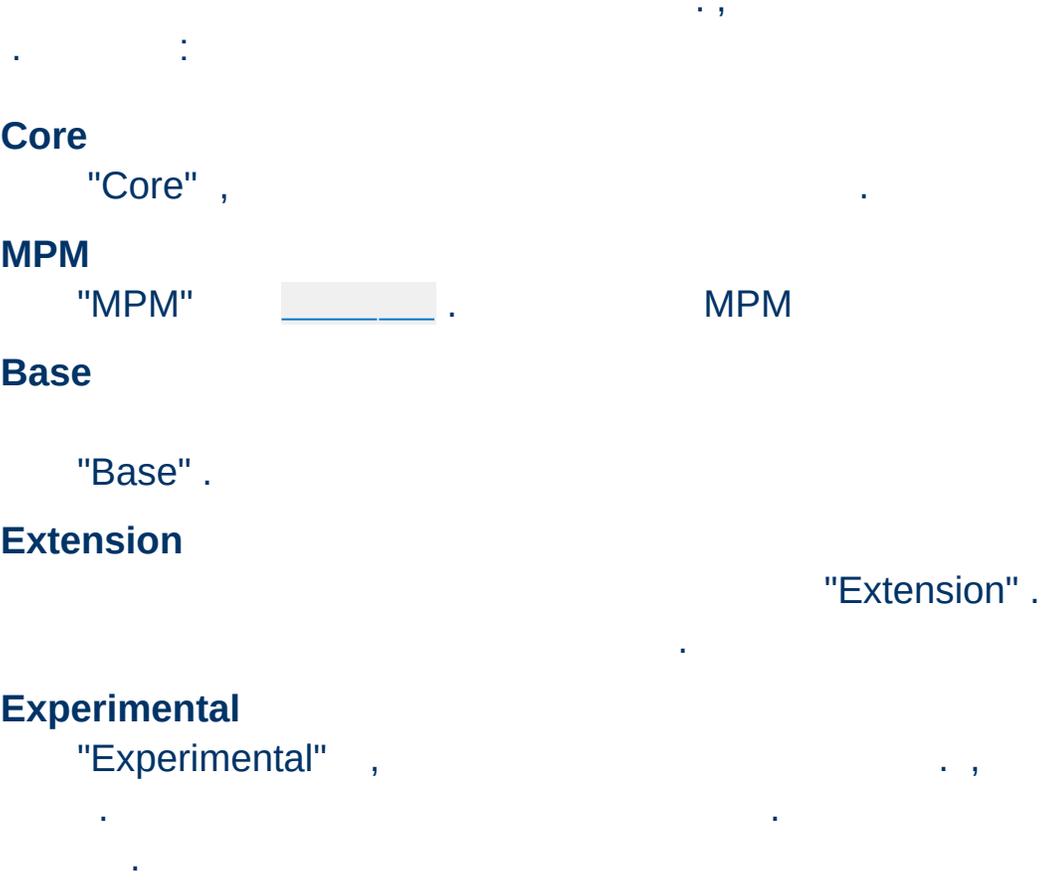
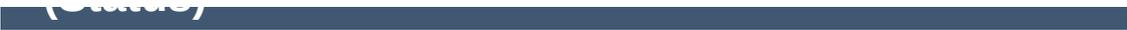


## Override (Override)

```
.htaccess          override .  
.htaccess          .
```

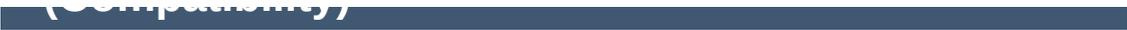
```
Overrides AllowOverride ,      (  
  AllowOverride .  
override .
```





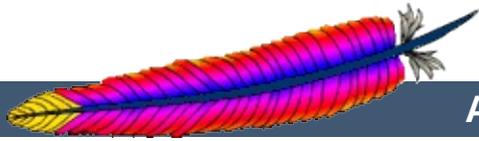
.





2 ,

..



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

## Apache Core Features

**Description:** Core Apache HTTP Server features that are always available

**Status:** Core



## AcceptFilter Directive

<b>Description:</b>	Configures optimizations for a Protocol's Listener Sockets
<b>Syntax:</b>	<code>AcceptFilter protocol accept_filter</code>
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache 2.1.5 and later

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](#) and Linux's more primitive `TCP_DEFER_ACCEPT` are currently supported.

The default values on FreeBSD are:

```
AcceptFilter http httppready
AcceptFilter https dataready
```

The `httppready` 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.

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 nntp:

```
AcceptFilter nntp none
```

## 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
<b>Compatibility:</b>	Available in Apache 2.0.30 and later

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 MIME media 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](#)



## AddOutputFilterByType Directive

<b>Description:</b>	assigns an output filter to a particular MIME-type
<b>Syntax:</b>	AddOutputFilterByType <i>filter</i> [ <i>;filter...</i> ] <i>MIME-type</i> [ <i>MIME-type</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>	Available in Apache 2.0.33 and later; deprecated in Apache 2.1 and later

This directive activates a particular output [filter](#) for a request depending on the response [MIME-type](#). Because of certain problems discussed below, this directive is deprecated. The same functionality is available using [mod\\_filter](#).

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/plain
```

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 text/html
</Location>
```

## Note

Enabling filters with [AddOutputFilterByType](#) may fail partially or completely in some cases. For example, no filters are applied if the [MIME-type](#) could not be determined and falls back to the [DefaultType](#) setting, even if the [DefaultType](#) is the same.

However, if you want to make sure, that the filters will be applied, assign the content type to a resource explicitly, for example with [AddType](#) or [ForceType](#). Setting the content type within a (non-nph) CGI script is also safe.

## See also

- [AddOutputFilter](#)
- [SetOutputFilter](#)
- [filters](#)



<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>	Available in Apache httpd 2.0.46 and later. NoDecode option available in 2.2.18 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 <code>.htaccess</code> files
<b>Syntax:</b>	<code>AllowOverride All None directive-type [directive-type] ...</code>
<b>Default:</b>	<code>AllowOverride All</code>
<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`, then `.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.

### AuthConfig

Allow use of the authorization directives  
[\(AuthDBMGroupFile, AuthDBMUserFile,](#)  
[AuthGroupFile, AuthName, AuthType, AuthUserFile,](#)

[Require](#), etc.).

## FileInfo

Allow use of the directives controlling document types ([DefaultType](#), [ErrorDocument](#), [ForceType](#), [LanguagePriority](#), [SetHandler](#), [SetInputFilter](#), [SetOutputFilter](#), and [mod\\_mime](#) Add\* and Remove\* directives, etc.), 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](#)).

## 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.

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](#)
- [Configuration Files](#)
- [.htaccess Files](#)



<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>	Core
<b>Module:</b>	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](#)



<b>Description:</b>	Type of user authentication
<b>Syntax:</b>	AuthType Basic Digest
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Core
<b>Module:</b>	core

This directive selects the type of user authentication for a directory. The authentication types available are Basic (implemented by [mod\\_auth\\_basic](#)) and Digest (implemented by [mod\\_auth\\_digest](#)).

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](#).

### See also

- [Authentication and Authorization](#)
- [Access Control](#)



<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 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 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.



## DefaultType Directive

<b>Description:</b>	MIME content-type that will be sent if the server cannot determine a type in any other way
<b>Syntax:</b>	DefaultType <i>MIME-type none</i>
<b>Default:</b>	DefaultType text/plain
<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 2.2.7 and later

There will be times when the server is asked to provide a document whose type cannot be determined by its [MIME types](#) mappings.

The server SHOULD inform the client of the content-type of the document. If the server is unable to determine this by normal means, it will set it to the configured `DefaultType`. For example:

```
DefaultType image/gif
```

would be appropriate for a directory which contained many GIF images with filenames missing the `.gif` extension.

In cases where it can neither be determined by the server nor the administrator (e.g. a proxy), it is preferable to omit the MIME type altogether rather than provide information that may be false. This can be accomplished using

```
DefaultType None
```

`DefaultType None` is only available in `httpd-2.2.7` and later.

Note that unlike [ForceType](#), this directive only provides the default mime-type. All other mime-type definitions, including filename extensions, that might identify the media type will override this default.



## 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>
```

Be careful with the *directory-path* arguments: They have to literally match the filesystem path which Apache 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}">
```

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 ~ "public_html/.*">
  # ... 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 Apache access for `<Directory />` is `Allow from All`. This means that Apache will serve any file mapped from an URL. It is recommended that you change this with a block such as**

```
<Directory />
  Order Deny,Allow
  Deny from All
</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 file-system directories matching a regular expression and their subdirectories
<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 *sub-directories of that 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}">
```

would match directories in `/www/` that consisted of three numbers.

### End-of-line character

The end-of-line character (\$) cannot be matched with this directive.

### 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://www.my.host.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>	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 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 On
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in version 2.0.44 and later

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 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 or SMB), the kernel may be unable to serve the network file through its

own cache.

For server configurations that are vulnerable to these problems, you should disable this feature by specifying:

```
EnableSendfile Off
```

For NFS or SMB 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_disk_cache`. Only global definition of `EnableSendfile` is taken into account by the module.



<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
<b>Compatibility:</b>	Quoting syntax for text messages is different in Apache 2.0

In the event of a problem or error, Apache 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 will sometimes offer additional information regarding the problem/error.

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. Examples:

```
ErrorDocument 500 http://foo.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.pl
```

```
ErrorDocument 401 /subscription_info.html  
ErrorDocument 403 "Sorry can't allow you access today"
```

Additionally, the special value `default` can be used to specify Apache's simple hardcoded message. While not required under normal circumstances, `default` will restore Apache'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 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 will not generate custom error documents for proxied content.

Prior to version 2.0, messages were indicated by prefixing them with a single unmatched double quote character.

## 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> ]
<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**.

### Example

```
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.

### Example

```
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)`.

### Example

```
ErrorLog syslog:user
```

**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 Log Files](#)



## 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 INode MTime Size
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core

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.) In Apache 1.3.22 and earlier, the ETag value was *always* formed from the file's inode, size, and last-modified time (mtime). 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 INode 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 <i>filename</i>&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 here
</Files>

<Files "?at.*">
    # This would apply to cat.html, bat.html, hat.php and so on.
</Files>
```

[Regular expressions](#) can also be used, with the addition of the ~ character. For example:

```
<Files ~ "\.(gif|jpe?g|png)$">
```

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)$">
```

would match most common Internet graphics formats.

## 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 MIME content-type
<b>Syntax:</b>	ForceType <i>MIME-type</i>  None
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Moved to the core in Apache 2.0

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 *MIME-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 unlike `DefaultType`, this directive overrides all mime-type associations, including filename extensions, that might identify the media type.

You can override any `ForceType` setting 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>
```



<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.



<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.

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.

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.

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.

[RFC 7231 §4.1](#) "Request Methods" "Overview" requires that origin servers shall respond with an error 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. 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.

[RFC 2616 §19.6](#) "Compatibility With Previous Versions" had encouraged HTTP servers to support legacy HTTP/0.9 requests. RFC 7230 supercedes 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.



<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.

`<IfDefine>` sections are nest-able, which can be used to implement simple multiple-parameter tests. Example:

```
httpd -DReverseProxy -DUseCache -DMemCache ...
```

```
# httpd.conf
<IfDefine ReverseProxy>
  LoadModule proxy_module modules/mod_proxy.so
  LoadModule proxy_http_module modules/mod_proxy_http.so
  <IfDefine UseCache>
    LoadModule cache_module modules/mod_cache.so
    <IfDefine MemCache>
      LoadModule mem_cache_module modules/mod_mem_cache.so
    </IfDefine>
    <IfDefine !MemCache>
      LoadModule disk_cache_module modules/mod_disk_cache.so
    </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 -- 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</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Wildcard matching available in 2.0.41 and later

This directive allows inclusion of other configuration files from within the server configuration files.

Shell-style (`fnmatch()`) wildcard characters can be used to include several files at once, in alphabetical order. In addition, if `Include` points to a directory, rather than a file, Apache will read all files in that directory and any subdirectory. But including entire directories is not recommended, because it is easy to accidentally leave temporary files in a directory that can cause [httpd](#) to fail.

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
```

**See also**

- [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 `MaxRequestsPerChild` 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>seconds</i>
<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 will wait for a subsequent request before closing the connection. 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.

In a name-based virtual host context, the value of the first defined virtual host (the default host) in a set of [NameVirtualHost](#) will be used. The other values will be ignored.



<b>Description:</b>	Restrict enclosed access controls to only certain HTTP methods
<b>Syntax:</b>	<code>&lt;Limit <i>method</i> [<i>method</i>] ... &gt; ...</code> <code>&lt;/Limit&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Core
<b>Module:</b>	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.

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.



<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>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<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
<b>Compatibility:</b>	Available in Apache 2.0.47 and later

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'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.

### Example

```
LimitInternalRecursion 5
```



<b>Description:</b>	Restricts the total size of the HTTP request body sent from the client
<b>Syntax:</b>	LimitRequestBody <i>bytes</i>
<b>Default:</b>	LimitRequestBody 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.

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

Note: not applicable to proxy requests.



<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 `NameVirtualHost` the connection was mapped to.



<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 for the `NameVirtualHost` the connection was mapped to.



<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 for the `NameVirtualHost` the connection was mapped to.



<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> ...
```

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/> ...
```

### 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">
```

would match URLs that contained the substring `/extra/data` or `/special/data`. The directive `<LocationMatch>` behaves identical to the regex version of `<Location>`.

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
  Order Deny,Allow
  Deny from all
  Allow from .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



**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">
```

would match URLs that contained the substring /extra/data or /special/data.

## 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>	Controls the verbosity of the ErrorLog
<b>Syntax:</b>	LogLevel <i>level</i>
<b>Default:</b>	LogLevel warn
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core

**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	"Opening config file ..."

messages
----------

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`.



## 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
```



## 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 2.2.21 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>	2.2.28 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 2.2.28, where trailers were always merged.



<b>Description:</b>	Designates an IP address for name-virtual hosting
<b>Syntax:</b>	<code>NameVirtualHost <i>addr[:port]</i></code>
<b>Context:</b>	server config
<b>Status:</b>	Core
<b>Module:</b>	core

The `NameVirtualHost` directive is a required directive if you want to configure [name-based virtual hosts](#).

Although `addr` can be hostname it is recommended that you always use an IP address and a port, e.g.

```
NameVirtualHost 111.22.33.44:80
```

With the `NameVirtualHost` directive you specify the IP address on which the server will receive requests for the name-based virtual hosts. This will usually be the address to which your name-based virtual host names resolve. In cases where a firewall or other proxy receives the requests and forwards them on a different IP address to the server, you must specify the IP address of the physical interface on the machine which will be servicing the requests. If you have multiple name-based hosts on multiple addresses, repeat the directive for each address.

### Note

Note, that the "main server" and any `_default_` servers will **never** be served for a request to a `NameVirtualHost` IP address (unless for some reason you specify `NameVirtualHost` but then don't define any `VirtualHosts` for that address).

Optionally you can specify a port number on which the name-

based virtual hosts should be used, e.g.

```
NameVirtualHost 111.22.33.44:8080
```

IPv6 addresses must be enclosed in square brackets, as shown in the following example:

```
NameVirtualHost [2001:db8::a00:20ff:fea7:ccea]:8080
```

To receive requests on all interfaces, you can use an argument of `*:80`, or, if you are listening on multiple ports and really want the server to respond on all of them with a particular set of virtual hosts, `*`

```
NameVirtualHost *:80
```

### Argument to `<VirtualHost>` directive

Note that the argument to the `<VirtualHost>` directive must exactly match the argument to the `NameVirtualHost` directive.

```
NameVirtualHost 1.2.3.4:80
<VirtualHost 1.2.3.4:80>
# ...
</VirtualHost>
```

## 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 All
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Core
<b>Module:</b>	core

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`. This is the default setting.

### **ExecCGI**

Execution of CGI scripts using `mod_cgi` is permitted.

### **FollowSymLinks**

The server will follow symbolic links in this directory.

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](#).

## **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.

### Warning

Mixing `Options` with a + or - with those without is not valid syntax and is likely to cause unexpected results.

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 `All`.



<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](#)



## 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.



## Require Directive

<b>Description:</b>	Selects which authenticated users can access a resource
<b>Syntax:</b>	Require <i>entity-name</i> [ <i>entity-name</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Core
<b>Module:</b>	core

This directive selects which authenticated users can access a resource. Multiple instances of this directive are combined with a logical "OR", such that a user matching any **Require** line is granted access. The restrictions are processed by authorization modules. Some of the allowed syntaxes provided by [mod\\_authz\\_user](#) 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.

Other authorization modules that implement require options include [mod\\_authnz\\_ldap](#), [mod\\_authz\\_dbm](#), and [mod\\_authz\\_owner](#).

**Require** must be accompanied by [AuthName](#) and [AuthType](#) directives, and directives such as [AuthUserFile](#) and [AuthGroupFile](#) (to define users and groups) in order to work correctly. Example:

AuthType Basic

```
AuthName "Restricted Resource"
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.

If **Require** is used together with the **Allow** or **Deny** directives, then the interaction of these restrictions is controlled by the **Satisfy** directive.

Multiple **Require** directives do operate as logical "OR", but some underlying authentication modules may require an explicit configuration to let authentication be chained to others. This is typically the case with **mod\_authnz\_ldap**, which exports the **AuthzLDAPAuthoritative** in that intent.

## Removing controls in subdirectories

The following example shows how to use the **Satisfy** directive to disable access controls in a subdirectory of a protected directory. This technique should be used with caution, because it will also disable any access controls imposed by **mod\_authz\_host**.

```
<Directory /path/to/protected/>
  Require user david
</Directory>
<Directory /path/to/protected/unprotected>
  # All access controls and authentication are disabled
  # in this directory
  Satisfy Any
  Allow from all
</Directory>
```

---

## See also

- [Authentication and Authorization](#)
- [Access Control](#)
- [Satisfy](#)
- [mod\\_authz\\_host](#)



<b>Description:</b>	Limits the CPU consumption of processes launched by Apache 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 children servicing requests, not the Apache children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked from the Apache 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 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 children servicing requests, not the Apache children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked from the Apache 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 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 children servicing requests, not the Apache children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked from the Apache 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>	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>	Core
<b>Module:</b>	core
<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
Order allow,deny
Allow from 192.168.1
Satisfy Any
```

Since version 2.0.51 [Satisfy](#) directives can be restricted to

particular methods by [<Limit>](#) and [<LimitExcept>](#) sections.

## See also

- [Allow](#)
- [Require](#)



<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; option Registry-Strict is available in Apache 2.0 and later

This directive is used to control how Apache finds the interpreter used to run CGI scripts. The default setting is `Script`. This causes Apache 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 falls back to the behavior of

the Script option.

For example, the registry setting to have a script with the .pl extension processed via perl would be:

```
HKEY_CLASSES_ROOT\.pl\Shell\ExecCGI\Command\Default) =>  
C:\Perl\bin\perl.exe -wT
```

### Security

Be careful when using ScriptInterpreterSource Registry with [ScriptAlias](#)'ed directories, because Apache 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 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.



**Description:** Email address that the server includes in error messages sent to the client

**Syntax:** `ServerAdmin email-address | URL`

**Context:** server config, virtual host

**Status:** Core

**Module:** 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!



**Description:** Alternate names for a host used when matching requests to name-virtual hosts

**Syntax:** `ServerAlias hostname [hostname] ...`

**Context:** virtual host

**Status:** Core

**Module:** 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.domain.com
ServerAlias server server2.domain.com server2
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 Virtual Host documentation](#)



<b>Description:</b>	Hostname and port that the server uses to identify itself
<b>Syntax:</b>	<code>ServerName [scheme://]fully-qualified-domain-name[:port]</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	In version 2.0, this directive supersedes the functionality of the <code>Port</code> directive from version 1.3.

The `ServerName` directive sets the request scheme, hostname and port that the server uses to identify itself. This is used when creating redirection URLs.

Additionally, `ServerName` is used (possibly in conjunction with `ServerAlias`) to uniquely identify a virtual host, when using [name-based virtual hosts](#).

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
```

If no `ServerName` is specified, then the server attempts to deduce the hostname by performing a reverse lookup on the IP address. 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.

## See also

- [Issues Regarding DNS and Apache](#)
- [Apache virtual host documentation](#)
- [UseCanonicalName](#)
- [UseCanonicalPhysicalPort](#)
- [NameVirtualHost](#)
- [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 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.

### Example

```
ServerRoot /home/httpd
```

### 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 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/2.0

#### **ServerTokens Min[imal]**

Server sends (e.g.): Server : Apache/2.0.41

#### **ServerTokens OS**

Server sends (e.g.): Server : Apache/2.0.41 (Unix)

#### **ServerTokens Full (or not specified)**

Server sends (e.g.): Server : Apache/2.0.41 (Unix)  
PHP/4.2.2 MyMod/1.2

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.

## 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
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Moved into the core in Apache 2.0

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 can override an earlier defined `SetHandler` directive by using the value `None`.

## 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>	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>	Core
<b>Module:</b>	core
<b>Compatibility:</b>	Available in Apache httpd 2.2.18 and later

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:** Amount of time the server will wait for certain events before failing a request

**Syntax:** Timeout *seconds*

**Default:** Timeout 300

**Context:** server config, virtual host

**Status:** Core

**Module:** core

The **Timeout** directive defines the length of time Apache will wait for I/O in various circumstances:

1. When reading data from the client, the length of time to wait for a TCP packet to arrive if the read buffer is empty.
2. When writing data to the client, the length of time to wait for an acknowledgement of a packet if the send buffer is full.
3. In **mod\_cgi**, the length of time to wait for output from a CGI script.
4. In **mod\_ext\_filter**, the length of time to wait for output from a filtering process.
5. In **mod\_proxy**, the default timeout value if **ProxyTimeout** is not configured.



## TRACE ENABLE DIRECTIVE

<b>Description:</b>	Determines the behaviour 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
<b>Compatibility:</b>	Available in Apache 1.3.34, 2.0.55 and later

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 64k (plus 8k 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 64k.



**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 must construct a *self-referential* URL -- that is, a URL that refers back to the same server. With `UseCanonicalName On` Apache 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 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/splat`, *without the trailing slash*, then Apache will redirect them to `http://www.domain.com/splat/`. If you have authentication enabled, this will cause the user to have to authenticate twice (once for `www` and once again for `www.domain.com` -- see [the FAQ on this subject for more](#)

[information](#)). But if `UseCanonicalName` is set Off, then Apache 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 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](#)



**Description:** Configures how the server determines its own name and port

**Syntax:** UseCanonicalPhysicalPort On|Off

**Default:** UseCanonicalPhysicalPort Off

**Context:** server config, virtual host, directory

**Status:** Core

**Module:** core

In many situations Apache must construct a *self-referential* URL -- that is, a URL that refers back to the same server. With `UseCanonicalPhysicalPort On`, Apache 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 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 when the physical port is used is as follows:

`UseCanonicalName On`

- Port provided in Servername
- Physical port
- Default port

`UseCanonicalName Off | DNS`

- Parsed port from Host : header
- Physical port
- Port provided in Servername

- 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:

- 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 is used only in combination with `NameVirtualHost *` to match all IP addresses; or
- The string `_default_`, which is used only with IP virtual hosting to catch unmatched IP addresses.

### Example

```
<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]: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>
```

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 listens on. You may need to ensure that Apache is listening on the correct addresses using `Listen`.

When using IP-based virtual hosting, the special name `_default_` can be specified in which case this virtual host will match any IP address that is not explicitly listed in another virtual host. In the absence of any `_default_` virtual host the "main" server config, consisting of all those definitions outside any `VirtualHost` section, is used when no IP-match occurs. (But note that any IP address that matches a `NameVirtualHost` directive will use neither the "main" server config nor the `_default_` virtual host. See the [name-based virtual hosting](#) documentation for further details.)

You can specify a `:port` to change the port that is matched. If unspecified then it defaults to the same port as the most recent

[Listen](#) statement of the main server. You may also specify `:*` to match all ports on that address. (This is recommended when used with `_default_`.)

A [ServerName](#) should be specified inside each `<VirtualHost>` block. If it is absent, the [ServerName](#) from the "main" server configuration will be inherited.

### 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.

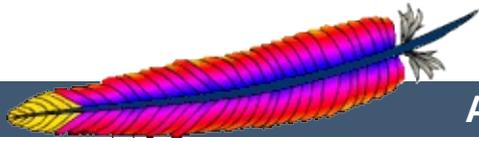
### See also

- [Apache Virtual Host documentation](#)
- [Issues Regarding DNS and Apache](#)
- [Setting which addresses and ports Apache 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

## Apache MPM Common Directives

**Description:** A collection of directives that are implemented by more than one multi-processing module (MPM)

**Status:** MPM



<b>Description:</b>	Method that Apache uses to serialize multiple children accepting requests on network sockets
<b>Syntax:</b>	AcceptMutex Default <i>method</i>
<b>Default:</b>	AcceptMutex Default
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">prefork</a> , <a href="#">worker</a>

The `AcceptMutex` directive sets the method that Apache uses to serialize multiple children accepting requests on network sockets. Prior to Apache 2.0, the method was selectable only at compile time. The optimal method to use is highly architecture and platform dependent. For further details, see the [performance tuning](#) documentation.

If this directive is set to `Default`, then the compile-time selected default will be used. Other possible methods are listed below. Note that not all methods are available on all platforms. If a method is specified which is not available, a message will be written to the error log listing the available methods.

### **flock**

uses the `flock(2)` system call to lock the file defined by the [LockFile](#) directive.

### **fcntl**

uses the `fcntl(2)` system call to lock the file defined by the [LockFile](#) directive.

### **posixsem**

uses POSIX compatible semaphores to implement the mutex.

### **pthread**

uses POSIX mutexes as implemented by the POSIX Threads

(PThreads) specification.

### **sysvsem**

uses SysV-style semaphores to implement the mutex.

If you want to find out the compile time chosen default for your system, you may set your [LogLevel](#) to debug. Then the default [AcceptMutex](#) will be written into the [ErrorLog](#).

#### **Warning**

On most systems, when the `pthread` option is selected, if a child process terminates abnormally while holding the `AcceptCntl` mutex the server will stop responding to requests. When this occurs, the server will require a manual restart to recover.

Solaris is a notable exception as it provides a mechanism, used by Apache, which usually allows the mutex to be recovered after a child process terminates abnormally while holding a mutex.

If your system implements the `pthread_mutexattr_setrobust_np()` function, you may be able to use the `pthread` option safely.



<b>Description:</b>	Directory for apache to run <i>chroot(8)</i> after startup.
<b>Syntax:</b>	<code>ChrootDir /path/to/directory</code>
<b>Default:</b>	none
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">event</a> , <a href="#">prefork</a> , <a href="#">worker</a>
<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.

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.



## CoreDumpDirectory Directive

<b>Description:</b>	Directory where Apache 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="#">beos</a> , <a href="#">mpm_winnt</a> , <a href="#">prefork</a> , <a href="#">worker</a>

This controls the directory to which Apache attempts to switch before dumping core. The default is in the [ServerRoot](#) directory, however since this should not be writable by the user the server runs as, core dumps won't normally get written. If you want a core dump for debugging, you can use this directive to place it in a different location.

### Core Dumps on Linux

If Apache starts as root and switches to another user, the Linux kernel *disables* core dumps even if the directory is writable for the process. Apache (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.



## EnableExceptionHook Directive

<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="#">prefork</a> , <a href="#">worker</a>
<b>Compatibility:</b>	Available in version 2.0.49 and later

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="#">prefork</a> , <a href="#">worker</a> , <a href="#">event</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>	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>	MPM
<b>Module:</b>	<a href="#">beos</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</a>
<b>Compatibility:</b>	Only valid in global server config since Apache 2.0

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.

---

Special note: Use of this directive in `<VirtualHost>` is no longer supported. To configure your server for `suexec` use `SuexecUserGroup`.

#### Note

Although the `Group` directive is present in the `beos` and `mpmt_os2` MPMs, it is actually a no-op there and only exists for compatibility reasons.



<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="#">beos</a> , <a href="#">mpm_network</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</a> , <a href="#">event</a>
<b>Compatibility:</b>	Required directive since Apache 2.0 The <i>protocol</i> argument was added in 2.1.5

The `Listen` directive instructs Apache 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.

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 uses](#)



<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="#">beos</a> , <a href="#">mpm_network</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</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>	Location of the accept serialization lock file
<b>Syntax:</b>	LockFile <i>filename</i>
<b>Default:</b>	LockFile logs/accept.lock
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">prefork</a> , <a href="#">worker</a>

The `LockFile` directive sets the path to the lockfile used when Apache is used with an `AcceptMutex` value of either `fcntl` or `flock`. This directive should normally be left at its default value. The main reason for changing it is if the `logs` directory is NFS mounted, since **the lockfile must be stored on a local disk**. The PID of the main server process is automatically appended to the filename.

### Security

It is best to *avoid* putting this file 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.

### See also

- [AcceptMutex](#)



<b>Description:</b>	Maximum number of connections that will be processed simultaneously
<b>Syntax:</b>	MaxClients <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">beos</a> , <a href="#">prefork</a> , <a href="#">worker</a>

The `MaxClients` directive sets the limit on the number of simultaneous requests that will be served. Any connection attempts over the `MaxClients` 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](#)), `MaxClients` 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. [beos](#) or [worker](#)) `MaxClients` restricts the total number of threads that will be available to serve clients. The default value for [beos](#) is 50. For hybrid MPMs the default value is 16 (`ServerLimit`) multiplied by the value of 25 (`ThreadsPerChild`). Therefore, to increase `MaxClients` to a value that requires more than 16 processes, you must also raise `ServerLimit`.



<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> 0
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">beos</a> , <a href="#">mpm_network</a> , <a href="#">prefork</a> , <a href="#">worker</a> , <a href="#">mpm_winnt</a>

The `MaxMemFree` directive sets the maximum number of free Kbytes that the main allocator is allowed to hold without calling `free()`. When not set, or when set to zero, the threshold will be set to unlimited.



## MaxRequestsPerChild Directive

<b>Description:</b>	Limit on the number of requests that an individual child server will handle during its life
<b>Syntax:</b>	MaxRequestsPerChild <i>number</i>
<b>Default:</b>	MaxRequestsPerChild 10000
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">mpm_netware</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</a>

The `MaxRequestsPerChild` directive sets the limit on the number of requests that an individual child server process will handle. After `MaxRequestsPerChild` requests, the child process will die. If `MaxRequestsPerChild` is 0, then the process will never expire.

### Different default values

The default value for [mpm\\_netware](#) and [mpm\\_winnt](#) is 0.

Setting `MaxRequestsPerChild` to a non-zero value limits the amount of memory that process can consume by (accidental) memory leakage.

### Note

For `KeepAlive` requests, only the first request is counted towards this limit. In effect, it changes the behavior to limit the number of *connections* per child.

### Default Configuration

The default (compiled-in) value of this setting (10000) is used when no `MaxRequestsPerChild` directive is present in the

configuration. Many default configurations provided with the server include "MaxRequestsPerChild 0" as part of the *default configuration*.



<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="#">beos</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a> , <a href="#">worker</a>

Maximum number of idle threads. Different MPMs deal with this directive differently.

For [worker](#), 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.

For [mpm\\_netware](#) the default is `MaxSpareThreads 100`. Since this MPM runs a single-process, the spare thread count is also server-wide.

[beos](#) and [mpmt\\_os2](#) work similar to [mpm\\_netware](#). The default for [beos](#) is `MaxSpareThreads 50`. For [mpmt\\_os2](#) the default value is 10.

## Restrictions

The range of the `MaxSpareThreads` value is restricted. Apache will correct the given value automatically according to the following rules:

- [mpm\\_netware](#) wants the value to be greater than `MinSpareThreads`.
- For [worker](#) the value must be greater or equal than the sum of `MinSpareThreads` and `ThreadsPerChild`.

---

## See also

- [MinSpareThreads](#)
- [StartServers](#)



<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="#">beos</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a> , <a href="#">worker</a>

Minimum number of idle threads to handle request spikes. Different MPMs deal with this directive differently.

[worker](#) uses 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.

[mpm\\_netware](#) uses a default of MinSpareThreads 10 and, since it is a single-process MPM, tracks this on a server-wide bases.

[beos](#) and [mpmt\\_os2](#) work similar to [mpm\\_netware](#). The default for [beos](#) is MinSpareThreads 1. For [mpmt\\_os2](#) the default value is 5.

### See also

- [MaxSpareThreads](#)
- [StartServers](#)



<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="#">beos</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</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 2 it is recommended to use only the [apachectl](#) script 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="#">beos</a> , <a href="#">mpm_netware</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</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="#">beos</a> , <a href="#">mpm_winnt</a> , <a href="#">prefork</a> , <a href="#">worker</a>

Apache 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 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 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](#)



<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="#">beos</a> , <a href="#">mpm_network</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">worker</a>

The server will set the TCP send buffer size to the number of bytes specified. Very useful to increase past standard OS defaults on high speed high latency (*i.e.*, 100ms or so, such as transcontinental fast pipes).

If set to the value of 0, the server will use the OS default.

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>	<code>ServerLimit</code> <i>number</i>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<code>prefork</code> , <code>worker</code>

For the `prefork` MPM, this directive sets the maximum configured value for `MaxClients` for the lifetime of the Apache process. For the `worker` MPM, this directive in combination with `ThreadLimit` sets the maximum configured value for `MaxClients` for the lifetime of the Apache process. Any attempts to change this directive during a restart will be ignored, but `MaxClients` 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 `MaxClients` are set to values higher than the system can handle, Apache may not start or the system may become unstable.

With the `prefork` MPM, use this directive only if you need to set `MaxClients` higher than 256 (default). Do not set the value of this directive any higher than what you might want to set `MaxClients` to.

With `worker` use this directive only if your `MaxClients` 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 `MaxClients` and `ThreadsPerChild`.

## 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.

## See also

- [Stopping and Restarting Apache](#)



<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="#">mpm_os2</a> , <a href="#">prefork</a> , <a href="#">worker</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, there is usually little reason to adjust this parameter.

The default value differs from MPM to MPM. For [worker](#) the default is `StartServers 3`. For [prefork](#) defaults to 5 and for [mpm\\_os2](#) to 2.



## StartThreads Directive

<b>Description:</b>	Number of threads created on startup
<b>Syntax:</b>	<code>StartThreads <i>number</i></code>
<b>Default:</b>	See usage for details
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">beos</a> , <a href="#">mpm_netware</a>

Number of threads created on startup. As the number of threads is dynamically controlled depending on the load, 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.

For [beos](#) the default is `StartThreads 10`. It also reflects 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="#">mpm_winnt</a> , <a href="#">worker</a>
<b>Compatibility:</b>	Available for <a href="#">mpm_winnt</a> in Apache 2.0.41 and later

This directive sets the maximum configured value for [ThreadsPerChild](#) for the lifetime of the Apache 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 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.

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 15000` with [mpm\\_winnt](#)) compiled into the

server. This is intended to avoid nasty effects caused by typos.



## 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="#">mpm_winnt</a> , <a href="#">worker</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>	<code>mpm_netware</code> , <code>mpm_winnt</code> , <code>worker</code> , <code>event</code>
<b>Compatibility:</b>	Available in Apache 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 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 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.

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.



<b>Description:</b>	The userid under which the server will answer requests
<b>Syntax:</b>	User <i>unix-userid</i>
<b>Default:</b>	User #-1
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<a href="#">prefork</a> , <a href="#">worker</a>
<b>Compatibility:</b>	Only valid in global server config since Apache 2.0

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.

Special note: Use of this directive in `<VirtualHost>` is no longer supported. To configure your server for `suexec` use `SuexecUserGroup`.

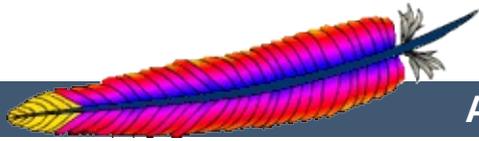
### Note

Although the **User** directive is present in the `beos` and `mpmt_os2` MPMs, it is actually a no-op there and only exists for compatibility reasons.

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## MPM beos

```
└─ BeOS
  └─ MPM
    └─ mpm_beos_module
      └─ beos.c
```

(MPM) BeOS .



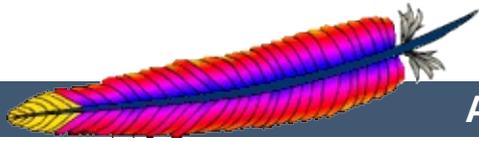
## MaxRequestsPerThread

```
MaxRequestsPerThread number
MaxRequestsPerThread 0
MPM
beos
```

```
MaxRequestsPerThread .
MaxRequestsPerThread .
MaxRequestsPerThread 0 .
MaxRequestsPerThread 0 :
```

- ( ) (memory leakage) ;
- .

```
KeepAlive .
```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

## Apache MPM event

<b>Description:</b>	An experimental variant of the standard <a href="#">worker</a> MPM
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_event_module
<b>Source File:</b>	event.c

### Summary

#### Warning

This MPM is experimental, so it may or may not work as expected.

The [event](#) Multi-Processing Module (MPM) is designed to allow more requests to be served simultaneously by passing off some processing work to supporting threads, freeing up the main threads to work on new requests. It is based on the [worker](#) MPM, which implements a hybrid multi-process multi-threaded server. Run-time configuration directives are identical to those provided by [worker](#).

To use the [event](#) MPM, add `--with-mpm=event` to the [configure](#) script's arguments when building the [httpd](#).

### See also

[The worker MPM](#)



---

This MPM tries to fix the 'keep alive problem' in HTTP. After a client completes the first request, the client can keep the connection open, and send further requests using the same socket. This can save significant overhead in creating TCP connections. However, Apache 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 thread to handle both the Listening sockets, and all sockets that are in a Keep Alive state.

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.

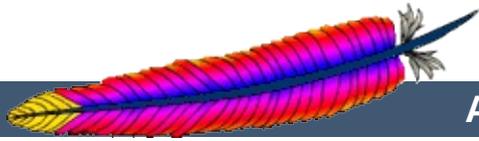
This MPM does not perform well on older platforms which lack good threading, but the requirement for EPoll or KQueue makes this moot.

- 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.

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 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 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.

[MaxRequestsPerChild](#) 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.

## See also

[Setting which addresses and ports Apache 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

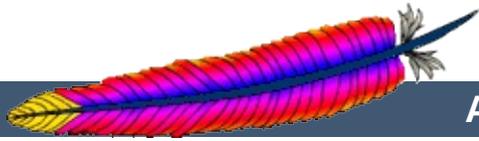
The **MaxThreads** directive sets the desired maximum number worker threads allowable. The default value is also the compiled in hard limit. Therefore it can only be lowered, for example:

```
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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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's 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 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](#).

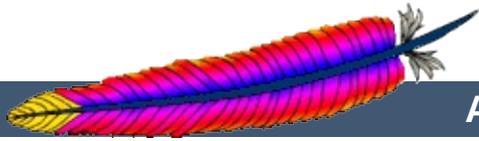
## See also

[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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 that handles requests in a manner similar to Apache 1.3. 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 **MaxClients** 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.

## See also

[Setting which addresses and ports Apache uses](#)



A single control process is responsible for launching child processes which listen for connections and serve them when they arrive. Apache 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 [MaxClients](#) regulate how the parent process creates children to serve requests. In general, Apache 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 [MaxClients](#), while sites with limited memory may need to decrease [MaxClients](#) 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 as a less-privileged user. The [User](#) and [Group](#) directives are used to set the privileges of the Apache 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.

[MaxRequestsPerChild](#) controls how frequently the server recycles processes by killing old ones and launching new ones.



<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 will automatically adjust it to `MinSpareServers + 1`.

## See also

- [MinSpareServers](#)
- [StartServers](#)



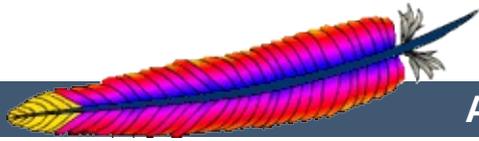
<b>Description:</b>	Minimum number of idle child server processes
<b>Syntax:</b>	<code>MinSpareServers</code> <i>number</i>
<b>Default:</b>	<code>MinSpareServers</code> 5
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	prefork

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 at a maximum rate of 1 per second.

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](#)



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

## Apache MPM winnt

<b>Description:</b>	This Multi-Processing Module is 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

### See also

[Using Apache HTTP Server on Microsoft Windows](#)

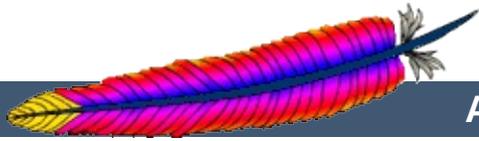


<b>Description:</b>	Use <code>accept()</code> rather than <code>AcceptEx()</code> to accept network connections
<b>Syntax:</b>	<code>Win32DisableAcceptEx</code>
<b>Default:</b>	<code>AcceptEx()</code> is enabled by default. Use this directive to disable use of <code>AcceptEx()</code>
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	<code>mpm_winnt</code>
<b>Compatibility:</b>	Available in Version 2.0.49 and later

`AcceptEx()` is a Microsoft WinSock v2 API that provides some performance improvements over the use of the BSD style `accept()` API in certain circumstances. Some popular Windows products, typically virus scanning or virtual private network packages, have bugs that interfere with the proper operation of `AcceptEx()`. If you encounter an error condition like:

```
[error] (730038)An operation was attempted on something that is not a socket.: winnt_accept: AcceptEx failed. Attempting to recover.
```

you should use this directive to disable the use of `AcceptEx()`.



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 [MaxClients](#), which controls the maximum total number of threads that may be launched.

### See also

[Setting which addresses and ports Apache uses](#)



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 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, Apache 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 [MaxClients](#) directive. The maximum number of active child processes is determined by the [MaxClients](#) 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 [MaxClients](#) 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. If non-default values are specified for these directives, they should appear before other [worker](#)

directives.

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 [MaxClients](#) 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 [MaxRequestsPerChild](#) to zero
- set the value of [MaxSpareThreads](#) to the same value as [MaxClients](#)

A typical configuration of the process-thread controls in the [worker](#) MPM could look as follows:

```
ServerLimit 16
StartServers 2
MaxClients 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 Apache as a less-privileged user. The [User](#) and [Group](#) directives are used to set the privileges of the Apache 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.

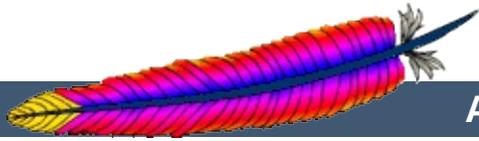
[MaxRequestsPerChild](#) controls how frequently the server

recycles processes by killing old ones and launching new ones.

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_actions



- ┆ CGI .
- ┆ Base
- ┆ actions\_module
- ┆ mod\_actions.c

┆            Action                    MIME content type CGI  
┆            Script                    CGI .  
┆

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

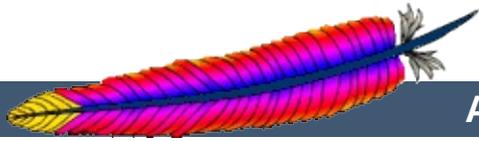
```
Script CGI .  
Script method cgi-script  
Script , , directory  
Script Base  
Script mod_actions
```

```
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](#) |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_alias

- , URL
- Base
- alias\_module
- mod\_alias.c

[ScriptAlias](#) URL . [DocumentRo](#)  
., [ScriptAlias](#) CGI .

[Redirect](#) URL .

[mod\\_alias](#) URL .

[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 .

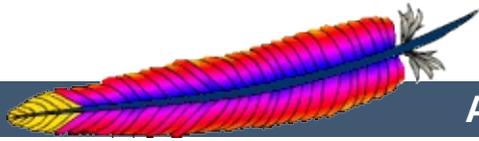


## ScriptAliasMatch

```
#: URL CGI
#: ScriptAliasMatch regex file-path|directory-
  path
#: ,
#: Base
#: mod_alias
```

```
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"
URLMatch /cgi-bin/ .
```

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_asis

[: HTTP](#)

[: Base](#)

[: asis\\_module](#)

[: mod\\_asis.c](#)

HTTP

cgi\_nph

HTTP

mime type      httpd/send-as-is .

[mod\\_headers](#)

[mod\\_cern\\_meta](#)



```
send-as-is .
```

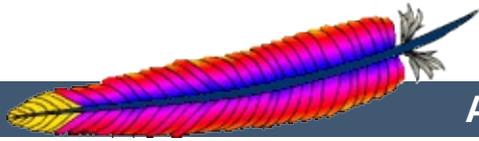
```
AddHandler send-as-is asis
```

```
.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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_auth\_basic

- Basic authentication
- Base
- auth\_basic\_module
- mod\_auth\_basic.c
- 2.1

(provider) HTTP Basic Authentication  
HTTP Digest Authentication mod\_auth\_digest.

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 .

```



```

:
: AuthBasicProvider On|Off|provider-name
: [i>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>

```

```

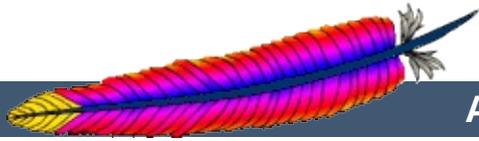
mod_authn_dbm mod_authn_file .

```

```

Off .

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_auth\_digest

└─ MD5 Digest Authentication

└─ Experimental

└─ auth\_digest\_module

└─ mod\_auth\_digest.c

HTTP Digest Authentication .

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

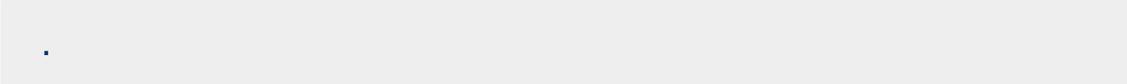
<a href="#">:</a>	digest authentication URI
<a href="#">:</a>	AuthDigestDomain <i>URI</i> [ <i>URI</i> ] ...
<a href="#">:</a>	directory, .htaccess
<a href="#">Override :</a>	AuthConfig
<a href="#">:</a>	Experimental
<a href="#">:</a>	mod_auth_digest

```
AuthDigestDomain ( / .URI . URI "" / .URI (scheme), , ) URL URI. , URI() Authorization . , AuthDigestNcC . URI , ( ) /
```



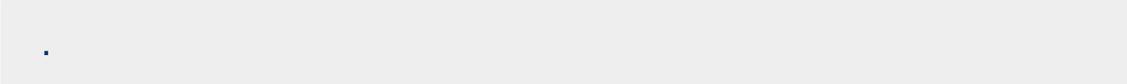
## AuthDigestNcCheck

```
: nonce-count
: AuthDigestNcCheck On|Off
: AuthDigestNcCheck Off
:
: Experimental
: mod_auth_digest
```



## AuthDigestNonceFormat

```
: nonce
: AuthDigestNonceFormat format
: directory, .htaccess
Override : AuthConfig
: Experimental
: mod_auth_digest
```



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>	.
<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 size
: AuthDigestShmemSize 1000
:
: Experimental
: mod_auth_digest

```

AuthDigestShmemSize

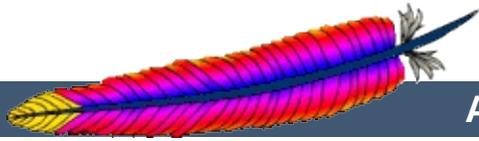
AuthDigestShmemSize 0

size , K M KBytes MBytes . , :

```

AuthDigestShmemSize 1048576
AuthDigestShmemSize 1024K
AuthDigestShmemSize 1M

```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

## Apache Module `mod_auth_alias`

<b>Description:</b>	Provides the ability to create extended authentication providers based on actual providers
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>auth_alias_module</code>
<b>Source File:</b>	<code>mod_auth_alias.c</code>
<b>Compatibility:</b>	Available in Apache 2.1 and later

### Summary

This module allows extended authentication providers to 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.



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

```
LoadModule authn_alias_module modules/mod_authn_alias.so

<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>
  Order deny,allow
```

```
Allow from all

AuthBasicProvider ldap-other-alias ldap-alias1

AuthType Basic
AuthName LDAP_Protected_Place
AuthzLDAPAuthoritative off
Require valid-user
</Directory>
```



**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>`

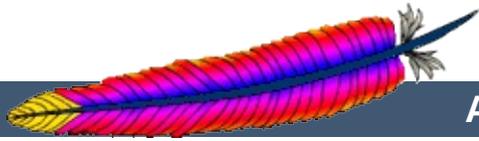
**Context:** server config

**Status:** Extension

**Module:** mod\_auth\_alias

`<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.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_auth\_anon

- :(anonymous)"
- :(Extension
- :(authn\_anon\_module
- :(mod\_auth\_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

```
:\n:\nAnonymous_LogEmail On|Off\n:\nAnonymous_LogEmail On\n:\ndirectory, .htaccess\nOverride : AuthConfig\n:\nExtension\n:\nmod_authn_anon
```

On ( ) " .



Anonymous\_MustGiveEmail

```
:  
: Anonymous_MustGiveEmail On|Off  
: Anonymous_MustGiveEmail On  
: directory, .htaccess  
Override : AuthConfig  
: Extension  
: mod_auth_anon
```

. .

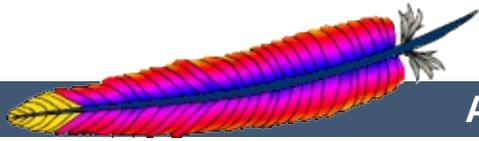


:	
:	Anonymous_NoUserID On Off
:	Anonymous_NoUserID Off
:	directory, .htaccess
<b>Override :</b>	AuthConfig
:	Extension
:	mod_authn_anon

On ( ) .  
OK MS-Explorer .







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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

# Apache Module `mod_authn_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.

## See also

- [AuthName](#)
- [AuthType](#)
- [AuthBasicProvider](#)
- [AuthDigestProvider](#)
- [DBDriver](#)
- [DBDParams](#)



## Configuration Example

This simple example shows use of this module in the context of the Authentication and DBD frameworks. Please note that you need to load an authorization module, such as [mod\\_authz\\_user](#), to get it working.

```
# mod_dbd configuration
DBDriver pgsql
DBDParams "dbname=apacheauth user=apache password=xxxxxx"

DBDMin 4
DBDKeep 8
DBDMax 20
DBDExptime 300

<Directory /usr/www/myhost/private>
# core authentication and mod_auth_basic configuration
# for mod_authn_dbd
AuthType Basic
AuthName "My Server"
AuthBasicProvider dbd

# core authorization configuration
Require valid-user

# mod_authn_dbd SQL query to authenticate a user
AuthDBDUserPWQuery \
    "SELECT password FROM authn WHERE user = %s"
</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.

### Example

```
AuthDBDUserPWQuery \  
"SELECT password FROM authn WHERE user = %s"
```

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.



**Description:** SQL query to look up a password hash for a user and realm.

**Syntax:** AuthDBDUserRealmQuery *query*

**Context:** directory

**Status:** Extension

**Module:** mod\_authn\_dbd

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.

### Example

```
AuthDBDUserRealmQuery \  
    "SELECT password FROM authn WHERE user = %s AND realm = %s"
```

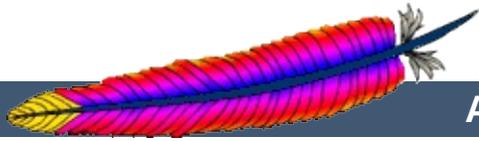
The first column value of the first row returned by the query statement should be a string containing the encrypted password. Subsequent rows will be ignored. If no rows are returned, the user will not be authenticated through `mod_authn_dbd`.

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.

Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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 .
```

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* .

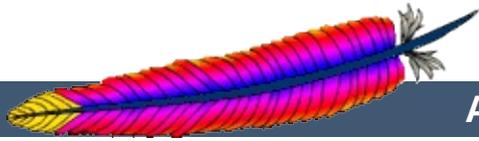
```

:
AuthDBMUserFile .
, AuthDBMUserFile .

```

: dbmopen NULL DBM  
. Netscape NULL  
.

[dbmmanage](#) Perl . DBM



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_authn\_default

- ┆
- ┆ Base
- ┆ authn\_default\_module
- ┆ mod\_authn\_default.c
- ┆ 2.1

[mod\\_auth\\_basic](#)



```

:
: AuthDefaultAuthoritative On|Off
: AuthDefaultAuthoritative On
: directory, .htaccess
Override : AuthConfig
: Base
: mod_authn_default

```

```

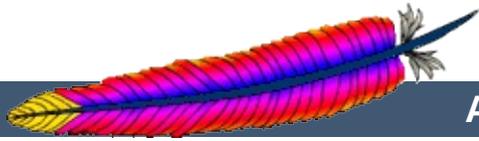
AuthDefaultAuthoritative Off ( modules.c
) .

```

```

mod_authn_default .
AuthDefaultAuthoritative ( On) .

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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 .
```

```
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
```

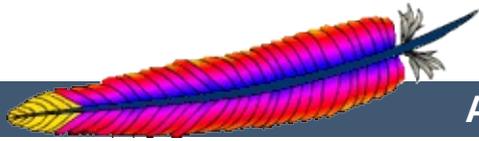
```
AuthUserFile
```

, `AuthUserFile` .

---

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

| | [FAQ](#) | |



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 provides 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.

## See also

[mod\\_ldap](#)  
[mod\\_auth\\_basic](#)  
[mod\\_authz\\_user](#)  
[mod\\_authz\\_groupfile](#)



- [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 Microsoft FrontPage with mod\\_authz\\_ldap](#)
  - [How It Works](#)
  - [Caveats](#)



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 the 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.
- 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. Note that if you use a [Require](#) value from another authorization module, you will need to ensure that [AuthzLDAPAuthoritative](#) is set to off to allow the authorization phase to fall back to the module providing the alternate [Require](#) value. When no LDAP-specific [Require](#) directives are used, authorization is allowed to fall back to other modules as if [AuthzLDAPAuthoritative](#) was set to off.

- 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.

AuthLDAPGroupAttribute

Determines the attribute to use for comparisons in the Require ldap-group directive.

AuthLDAPGroupAttributeIsDN

Specifies whether to use the user DN or the username when doing comparisons for the Require ldap-group directive.



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.

## 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=Airius?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 bjensson 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=Airius
objectClass: groupOfUniqueNames
uniqueMember: cn=Barbara Jenson, o=Airius
uniqueMember: cn=Fred User, o=Airius
```

The following directive would grant access to both Fred and Barbara:

```
Require ldap-group cn=Administrators, o=Airius
```

Behavior of this directive is modified by the [AuthLDAPGroupAttribute](#) and [AuthLDAPGroupAttributeIsDN](#) 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=Airius
```

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" status=active
```

## 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=marketing)
```

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.airius.com:389/ou=People,
o=Airius?uid?sub?(objectClass=*)"
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.airius.com
ldap2.airius.com/ou=People, o=Airius"
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.airius.com/ou=People, o=Airius?
cn"
Require valid-user
```

- Grant access to anybody in the Administrators group. The users must authenticate using their UID.

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid
Require ldap-group cn=Administrators, o=Airius
```

- The next example assumes that everyone at Airius 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.airius.com/o=Airius?uid??  
(qpagePagerID=*)  
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.airius.com/o=Airius?uid??(|  
(qpagePagerID=*)(uid=jmanager))  
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 Login Information

When this module performs authentication, LDAP attributes specified in the `AuthLDAPUrl` directive are placed in environment variables with the prefix "AUTHENTICATE\_".

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.



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>	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 <i>off on</i>
<b>Default:</b>	AuthLDAPBindAuthoritative on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap
<b>Compatibility:</b>	Available in versions later than 2.2.14

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 2.2.25.

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 arguments
AuthLDAPBindPassword "exec:/path/to/otherProgram argument1"
```



<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 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 check for group membership
<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 group membership. Multiple attributes can be used by specifying this directive multiple times. If not specified, then [mod\\_authnz\\_ldap](#) uses the member and uniquemember attributes.



## AuthLDAPGroupAttributeIsDN Directive

<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=Airius. If this directive is set,

[mod\\_authnz\\_ldap](#) will check if the group has cn=Babs Jenson, o=Airius as a member. If this directive is not set, then [mod\\_authnz\\_ldap](#) will check if the group has bjenson as a member.



**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>	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
```

## 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.

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.

### **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.

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.airius.com/o=Airius?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.

When [AuthLDAPURL](#) is enabled in a particular context, but some other module has performed authentication for the request, the server will try to map the username to a DN during authorization regardless of whether or not LDAP-specific requirements are present. To ignore the failures to map a username to a DN during authorization, set [AuthzLDAPAuthoritative](#) to "off".



<b>Description:</b>	Prevent other authentication modules from authenticating the user if this one fails
<b>Syntax:</b>	AuthzLDAPAuthoritative on off
<b>Default:</b>	AuthzLDAPAuthoritative on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_authnz_ldap

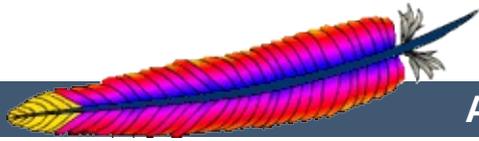
Set to off if this module should let other authorization modules attempt to authorize the user, should authorization with this module fail. Control is only passed on to lower modules if there is no DN or rule that matches the supplied user name (as passed by the client).

When no LDAP-specific [Require](#) directives are used, authorization is allowed to fall back to other modules as if [AuthzLDAPAuthoritative](#) was set to off.

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_authz\_dbm

- DBM
- Extension
- authz\_dbm\_module
- mod\_authz\_dbm.c
- 2.1

mod\_authz\_groupfile .

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 .

```
[ : ( ) ]
```



```

:
: AuthzDBMAuthoritative On|Off
: AuthzDBMAuthoritative On
: directory, .htaccess
Override : AuthConfig
: Extension
: mod_authz_dbm

```

```

AuthzDBMAuthoritative Off
( modules.c ) .
.

```

```

AuthAuthoritative .
Require
mod_authn_dbm mod_authn_file .
DBM , () .htpass
, .
.

```

```

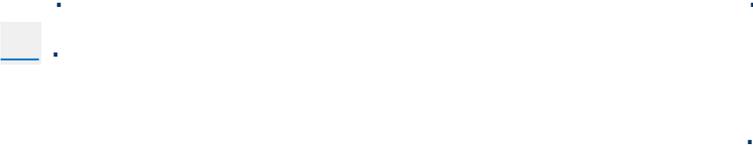
.htaccess ,
.htpasswd .

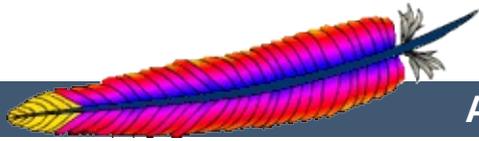
```



## AuthzDBMType

```
AuthzDBMType default|SDBM|GDBM|NDBM|DB
AuthzDBMType default
directory, .htaccess
Override : AuthConfig
Extension
mod_authz_dbm
```





| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_authz\_default

- ┆
- ┆ Base
- ┆ authz\_default\_module
- ┆ mod\_authz\_default.c
- ┆ 2.1

[mod\\_authz\\_user](#) [mod\\_authz\\_groupfile](#)



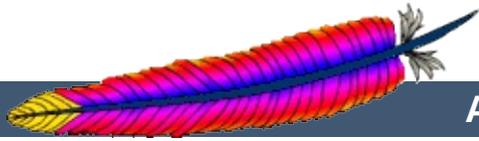
```

:
: AuthzDefaultAuthoritative On|Off
: AuthzDefaultAuthoritative On
: directory, .htaccess
Override : AuthConfig
: Base
: mod_authz_default

```

```
AuthzDefaultAuthoritative Off ( modules.c
) .
```

```
mod_authz_default
AuthzDefaultAuthoritative ( On) .
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_authz\_groupfile

- ┆
- ┆ Base
- ┆ authz\_groupfile\_module
- ┆ mod\_authz\_groupfile.c
- ┆ 2.1

Require  
Satisfy



## AuthGroupFile

```
:  
: AuthGroupFile file-path  
: directory, .htaccess  
Override : AuthConfig  
: Base  
: mod_authz_groupfile
```

### AuthGroupFile

```
. ServerRoot .  
'  
'
```

```
:  
mygroup: bob joe anne
```

```
. AuthDBMGroupFile  
.
```

```
AuthGroupFile .  
, AuthGroupFile .
```

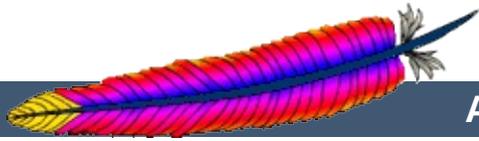


## AuthzGroupFileAuthenticating

```
:#
:# AuthzGroupFileAuthenticating On|Off
:# AuthzGroupFileAuthenticating On
:# directory, .htaccess
Override : AuthConfig
:# Base
:# mod_authz_groupfile
```

```
AuthzGroupFileAuthenticating Off
      (modules.c)
,
.
```

```
.htaccess ,
      .htpasswd .
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_authz\_host

- ( IP )
- Base
- authz\_host\_module
- mod\_authz\_host.c
- 2.1

```
<Directory>, <Files>, <Location>      .htaccess
    mod_authz_host . , IP ,
    Allow Deny ,
    Allow Deny .
    Satisfy
    ( GET, PUT, POST ) ,
<Limit>
```

Satisfy  
Require



## ALLOW

```
Allow from all|host|env=env-variable  
[host|env=env-variable] ...  
directory, .htaccess  
Override : Limit  
Base  
mod_authz_host
```

Allow ., IP, IP

from. . Allow from  
Deny Order  
host :

0

```
:  
Allow from apache.org
```

fooapache.org .  
HostnameLookups IP - DNS  
, IP DNS , IP

IP

```
:  
Allow from 10.1.2.3
```

IP

## IP

```
:
```

```
Allow from 10.1
```

IP 1 3

## /

```
:
```

```
Allow from 10.1.0.0/255.255.0.0
```

a.b.c.d w.x.y.z.

## /nnn CIDR

```
:
```

```
Allow from 10.1.0.0/16
```

, nnn 1

IPv6 IPv6

```
Allow from 2001:db8::a00:20ff:fea7:ccea  
Allow from 2001:db8::a00:20ff:fea7:ccea/10
```

**Allow** `variable`, `env-variable` `Allow from` `moc`

(), Referer, HTTP

```
:
```

```
SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in
<Directory /docroot>
  Order Deny,Allow
  Deny from all
  Allow from env=let_me_in
</Directory>
```

user-agent      KnockKnock/2.0      ,

.



## Deny

```
:  
: Deny from all|host|env=env-variable  
[host|env=env-variable] ...  
: directory, .htaccess  
Override : Limit  
: Base  
: mod_authz_host
```

, IP , .



```

: Allow
Deny .
: Order ordering
: Order Deny,Allow
: directory, .htaccess
Override : Limit
: Base
: mod_authz_host

```

Order Allow Deny

**Deny, Allow**

Deny Allow . .

**Allow, Deny**

Allow Deny . . .

Allow .

**Mutual-failure**

Deny Allow . 0

; . Allow Deny .

apache.org , .

```

Order Deny,Allow
Deny from all
Allow from apache.org

```

foo.apache.org , apache.org

. apache.org .

```
Order Allow,Deny
Allow from apache.org
Deny from foo.apache.org
```

Order Deny,Allow ,  
Allow from apache.org Deny from  
foo.apache.org . , apa

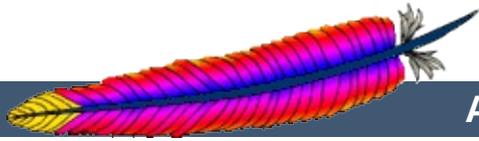
Order Allow De

```
<Directory /www>
  Order Allow,Deny
</Directory>
```

/www .

Order  
<Location> Allow Deny <Directory>  
.htaccess Allow Deny .

[Directory, Location, Files](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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" .
```

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>
```



```

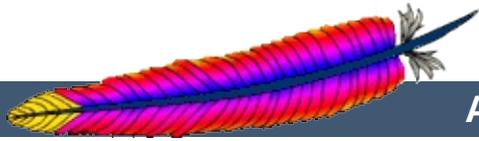
:
: AuthzOwnerAuthoritative On|Off
: AuthzOwnerAuthoritative On
: directory, .htaccess
Override : AuthConfig
: Extension
: mod_authz_owner

```

AuthzOwnerAuthoritative Off  
(modules.c )

- file-owner
- file-group

, Off file-owner file-group ,  
.  
,  
NCSA .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_authz\_user

- ┆
- ┆ Base
- ┆ authz\_user\_module
- ┆ mod\_authz\_user.c
- ┆ 2.1

,  
Require user . , re  
user .

Require  
Satisfy



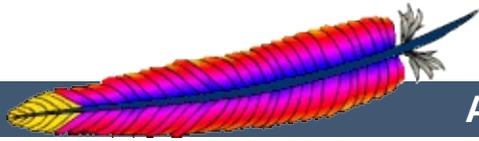
```

:
: AuthzUserAuthoritative On|Off
: AuthzUserAuthoritative On
: directory, .htaccess
Override : AuthConfig
: Base
: mod_authz_user

```

AuthzUserAuthoritative Off  
(modules.c ) .

,  
NCSA .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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 .



## AddIconByEncoding

```

: MIME content-encoding
: AddIconByEncoding icon MIME-encoding
  [MIME-encoding] ...
: , , directory, .htaccess
Override : Indexes
: Base
: mod_autoindex

```

```

FancyIndexing . Icon (%-esci
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](#)

.



<a href="#">:</a>	Inserts text in the HEAD section of an index page.
<a href="#">:</a>	IndexHeadInsert " <i>markup . . .</i> "
<a href="#">:</a>	, , directory, .htaccess
<b><a href="#">Override :</a></b>	Indexes
<a href="#">:</a>	Base
<a href="#">:</a>	mod_autoindex
<a href="#">:</a>	Available in Apache 2.2.11 and later

The documentation for this directive has not been translated yet. Please have a look at the English version.



## IndexIgnore

```
IndexIgnore file [file] ...  
IndexIgnore , , directory, .htaccess  
Override : Indexes  
Base  
mod_autoindex
```

**IndexIgnore** . *File*  
. IndexIgnore .  
. ( ) .

```
IndexIgnore README .htaccess *.bak *~
```



## 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

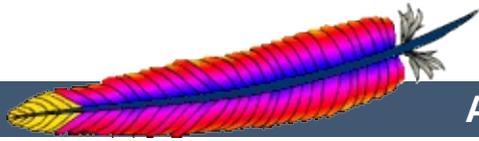
```
ReadmeName filename
, , directory,
.htaccess
Override : Indexes
Base
mod_autoindex
```

ReadmeName . *Filename* *Filename* DocumentRoot .

```
ReadmeName FOOTER.html
```

```
2
ReadmeName /include/FOOTER.html
```

HeaderName .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_cache

- URI
- Experimental
- cache\_module
- mod\_cache.c

mod\_cache

[RFC 261](#)

mod\_cache (storage management module)

mod\_disk\_cache

mod\_mem\_cache

mod\_mem\_cache

*proxy*) [ProxyPass](#) mod\_mem\_cache , (mod\_proxy)

URI



<u>mod disk cache</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 disk_cache_module modules/mod_disk_cache.so
  <IfModule mod_disk_cache.c>
    CacheRoot c:/cachertoot
    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
```



## 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\_disk\_cache .  
cache\_type fd mod\_mem\_cache .

( ) URL CacheEnable  
CacheEnable .

```
CacheEnable mem /manual  
CacheEnable fd /images  
CacheEnable disk /
```



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
```



## CacheIgnoreQueryString

```
: Ignore query string when caching
: CacheIgnoreQueryString On|Off
: CacheIgnoreQueryString Off
: ,
: Extension
: mod_cache
: Available in Apache 2.2.6 and later
```

The documentation for this directive has not been translated yet. Please have a look at the English version.



```
[:] Ignore defined session identifiers encoded in the URL when
caching
[:] CacheIgnoreURLSessionIdentifiers identifier
[identifier] ...
[:] CacheIgnoreURLSessionIdentifiers None
[:] ,
[:] Extension
[:] mod_cache
```

The documentation for this directive has not been translated yet.  
Please have a look at the English version.



```
┆ 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.
- [\[:\]](#) CacheLock *on|off*
- [\[:\]](#) CacheLock off
- [\[:\]](#) ,
- [\[:\]](#) Extension
- [\[:\]](#) mod\_cache
- [\[:\]](#) Available in Apache 2.2.15 and later

The documentation for this directive has not been translated yet.  
Please have a look at the English version.



```
: Set the maximum possible age of a cache lock.  
: CacheLockMaxAge integer  
: CacheLockMaxAge 5  
:  
: Extension  
: mod_cache
```

The documentation for this directive has not been translated yet. Please have a look at the English version.



```

: Set the lock path directory.
: CacheLockPath directory
: CacheLockPath /tmp/mod_cache-lock
: ,
: Extension
: mod_cache

```

The documentation for this directive has not been translated yet.  
Please have a look at the English version.



## CacheMaxExpire

```
CacheMaxExpire seconds  
CacheMaxExpire 86400 ()  
,  
Experimental  
mod_cache
```

CacheMaxExpire

HTTP

,

.

CacheMaxExpire 604800



```
[:] Attempt to cache requests or responses that have been
marked as no-store.
[:] CacheStoreNoStore On|Off
[:] CacheStoreNoStore Off
[:] ,
[:] Extension
[:] mod_cache
```

The documentation for this directive has not been translated yet.  
Please have a look at the [English version](#).

- [CacheIgnoreCacheControl](#)
- [CacheStorePrivate](#)



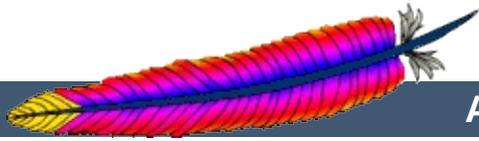
```

: Attempt to cache responses that the server has marked as
  private
: CacheStorePrivate On|Off
: CacheStorePrivate Off
: ,
: Extension
: mod_cache

```

The documentation for this directive has not been translated yet.  
Please have a look at the [English version](#).

- [CacheIgnoreCacheControl](#)
- [CacheStoreNoStore](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_cern\_meta

- ┆ CERN
- ┆ Extension
- ┆ cern\_meta\_module
- ┆ mod\_cern\_meta.c

CERN .

, Expires:

CERN .

HTTP

[CERN metafile semantics](#) .

[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

.



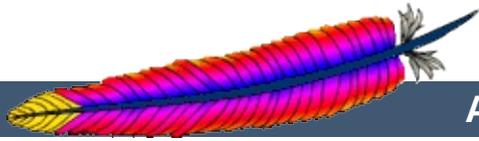
<a href="#">:</a>	CERN
<a href="#">:</a>	MetaSuffix <i>suffix</i>
<a href="#">:</a>	MetaSuffix .meta
<a href="#">:</a>	, , directory, .htaccess
<b><a href="#">Override :</a></b>	Indexes
<a href="#">:</a>	Extension
<a href="#">:</a>	mod_cern_meta

. ,  
 DOCUMENT\_ROOT/somedir/index.html  
 DOCUMENT\_ROOT/somedir/.web/index.html.meta  
 MIME .

```

:
MetaSuffix .meta

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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 .
```

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 ,

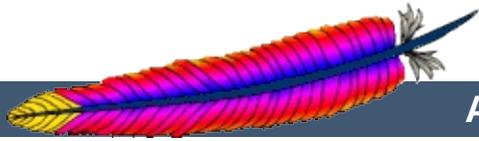
PUT POST



## ScriptLogLength

```
ScriptLogLength CGI
ScriptLogLength bytes
ScriptLogLength 10385760
,
Base
mod_cgi, mod_cgid
```

```
ScriptLogLength CGI .CGI (
)
CGI .
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



- [:~](#) 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.

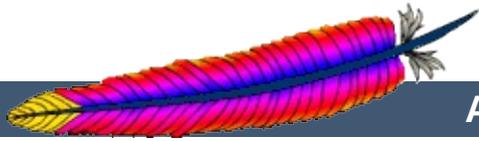


## ScriptSock

```
ScriptSock cgi  
ScriptSock file-path  
ScriptSock logs/cgisock  
,  
Base  
mod_cgid
```

```
CGI . (root)  
. CGI .
```

```
ScriptSock /var/run/cgid.sock
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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

.

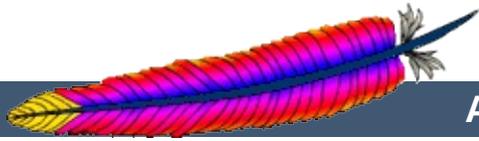
. iconv

```

<Directory /export/home/trawick/apacheinst/htdocs/convert>
  CharsetSourceEnc UTF-16BE
  CharsetDefault ISO-8859-1
</Directory>

```

## Solaris 8 iconv



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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

[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)

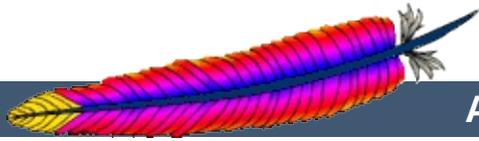
**DavMinTimeout**  
Folders 120 .

() . Microsoft Web  
**DavMinTimeout** (600 )

```

<Location /MSWord>
  DavMinTimeout 600
</Location>

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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 .

[mod\\_dav](#)



```

: DAV
: DavLockDB file-path
: ,
: Extension
: mod_dav_fs

```

**DavLockDB**

mod\_dav\_fs SDBM

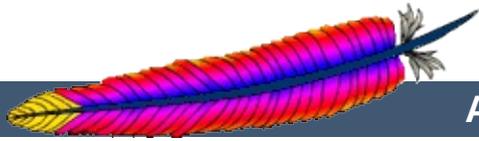
```

DavLockDB var/DavLock

```

User Group

DavLock



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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`.

## 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

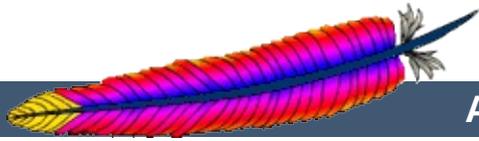
```
DavGenericLockDB var/DavLock
```

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

## 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.



[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*, server_rec*);

/* release a connection acquired with ap_dbd_open */
AP_DECLARE(void) ap_dbd_close(server_rec*, ap_dbd_t*);

/* acquire a connection that will have the lifetime of a request
 * and MUST NOT be explicitly closed. Return NULL on error.
 * This is the preferred function for most applications.
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_acquire(request_rec*);

/* acquire a connection that will have the lifetime of a connection
 * and MUST NOT be explicitly closed. Return NULL on error.
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_cacquire(conn_rec*);

/* Prepare a statement for use by a client module */
AP_DECLARE(void) ap_dbd_prepare(server_rec*, const char*, const c

/* Also export them as optional functions for modules that prefer
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_open, (apr_pool_t*, ser
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_close, (server_rec*, ap_dbd_
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_acquire, (request_rec*)
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_cacquire, (conn_rec*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepare, (server_rec*, const
```



## 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>	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 - see SECURITY note)**

username, password, appname, dbname, host, charset, lang, server

**MySQL**

host, port, user, pass, dbname, sock, flags, fldsz, group, reconnect

**ODBC**

datasource, user, password, connect, ctimeout, stimeout, access, txmode, bufsize

**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`



<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.



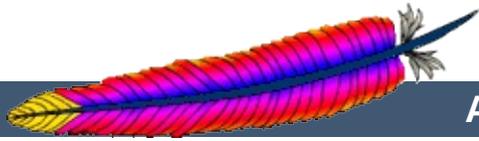
<b>Description:</b>	Specify an SQL driver
<b>Syntax:</b>	<code>DBDriver <i>name</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_dbd</code>

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_deflate

- 
- 
- Extension
- deflate\_module
- mod\_deflate.c

mod\_deflate

DE



## 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
- [:](#) 2.2.28 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
- `:` 2.2.28 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
: 2.2.28 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

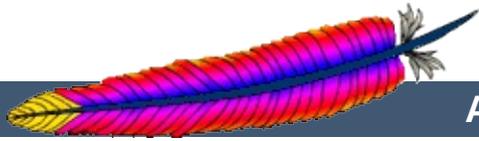


```

: Zlib window size
: DeflateWindowSize value
: DeflateWindowSize 15
: ,
: Extension
: mod_deflate

```

`DeflateWindowSize` zlib window size (1 15 )  
window size .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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/ .



## 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 .



## 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 .

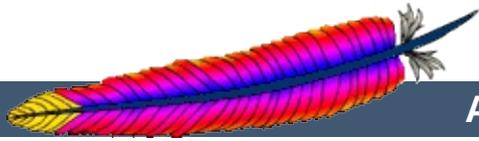
```
# !
<Location /some/path>
  DirectorySlash Off
  SetHandler some-handler
</Location>
```

```
(Options +Indexe
mod_autoindex DirectoryIndex (index.html )
URL .
index.html .
```



<a href="#">:</a>	Define a default URL for requests that don't map to a file
<a href="#">:</a>	FallbackResource disabled   <i>local-url</i>
<a href="#">:</a>	None - httpd will return 404 (Not Found)
<a href="#">:</a>	, , directory, .htaccess
<b><a href="#">Override :</a></b>	Indexes
<a href="#">:</a>	Base
<a href="#">:</a>	mod_dir
<a href="#">:</a>	Apache HTTP Server 2.2.16 and later - The disabled argument is supported since 2.2.24

The documentation for this directive has not been translated yet. Please have a look at the English version.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_disk\_cache

. .

- Content cache storage manager keyed to URIs
- Experimental
- disk\_cache\_module
- mod\_disk\_cache.c

. ...

mod disk cache .

mod

URI . .

:  
mod disk cache mod cache .



## CacheDirLength

```
CacheDirLength length  
CacheDirLength 2  
,  
Experimental  
mod_disk_cache
```

CacheDirLength .

CacheDirLevels CacheDirLength 20 .

CacheDirLength 4



## CACHEDIRLEVELS

```
CacheDirLevels levels
CacheDirLevels 3
Experimental
mod_disk_cache
```

```
CacheDirLevels .
```

```
CacheDirLevels CacheDirLength 20 .
```

```
CacheDirLevels 5
```





```
⋮  ()
⋮  CacheMinFileSize bytes
⋮  CacheMinFileSize 1
⋮  ,
⋮  Experimental
⋮  mod_disk_cache
```

CacheMinFileSize .

```
CacheMinFileSize 64
```

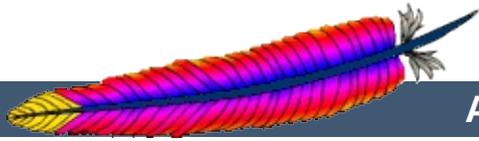


## CACHEROOT

```
└─┬─ root
   │
   └─ CacheRoot directory
      └─ ,
         └─ Experimental
            └─ mod_disk_cache
```

```
CacheRoot . mod_d
CacheDirLevels CacheDirLength root
CacheRoot
```

```
CacheRoot c:/cacheroor
```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 sample support

To enable the module, it should be compiled and loaded in to your running Apache configuration. Logging can then be enabled or disabled via the below directives.



## 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
```



## DumpIOLogLevel Directive

<b>Description:</b>	Controls the logging level of the DumpIO output
<b>Syntax:</b>	DumpIOLogLevel <i>level</i>
<b>Default:</b>	DumpIOLogLevel debug
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOLogLevel is only available in Apache 2.2.4 and later.

Enable dumping of all output at a specific [LogLevel](#) level.

### Example

```
DumpIOLogLevel notice
```

### Compatibility

Prior to 2.2.4 [mod\\_dumpio](#) would only dump to the log when [LogLevel](#) was set to debug



## 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.

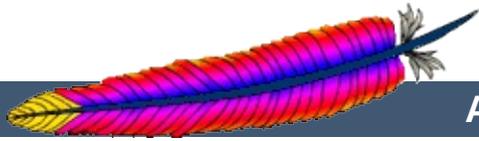
### Example

```
DumpIOOutput On
```

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_echo



- [:](#) echo
- [:](#) Experimental
- [:](#) echo\_module
- [:](#) mod\_echo.c
- [:](#) Apache 2.0

.  
,

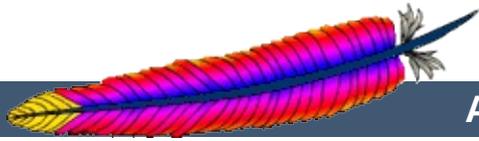
echo . telnet



```
ProtocolEcho
: echo
: ProtocolEcho On|Off
: ,
: Experimental
: mod_echo
: ProtocolEcho 2.0
.
```

ProtocolEcho echo .

```
ProtocolEcho On
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_env

.

- CGI SSI
- Base
- env\_module
- mod\_env.c

CGI SSI

.



## 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
```

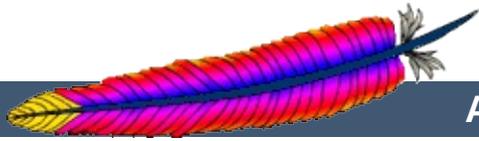


## UNSETENV

```
UnsetEnv env-variable [env-variable]
...
, , directory, .htaccess
Override : FileInfo
Base
mod_env
```

CGI SSI .

```
UnsetEnv LD_LIBRARY_PATH
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_example

- API
- Experimental
- example\_module
- mod\_example.c

```
modules/experimental      API
.
mod_example.c (callback)  .
. !
example .                  "example-handler"
example .
```



## Example

example :

1. `--enable-example configure .`
2. `(" make").`
- :
- A. `cp modules/experimental/mod_example.c  
modules/new_module/mod_myexample.c`
- B. `.`
- C. `modules/new_module/config.m4 .`
  1. `APACHE_MODPATH_INIT(new_module) .`
  2. `modules/experimental/config.m4 "example"  
APACHE_MODULE .`
  3. `"example" myexample .`
  4. `. cor`
  5. `C , CFLAGS,  
LDFLAGS, LIBS . modules confi`
  6. `APACHE_MODPATH_FINISH .`
- D. `module/new_module/Makefile.in .`  
`, include $(top_srcdir)/build/specia`
- E. `./buildconf .`
- F. `--enable-myexample`



example httpd.conf :

```
<Location /example-info>  
SetHandler example-handler  
</Location>
```

[.htaccess](#) , "test.example"

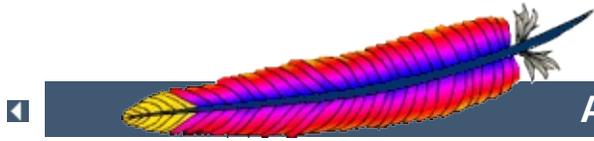
```
AddHandler example-handler .example
```



## Example

```
API
Example
, , directory,
.htaccess
Experimental
mod_example
```

```
Example example
example URL
. "Example directive declared here:
YES/NO"
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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

```
: MIME type Expires
: ExpiresByType MIME-type <code>seconds
: , , directory, .htaccess
Override : Indexes
: Extension
: mod_expires
```

```
( , text/html) Expires Cache-
Control max-age .
. Cache-Control: max-age ,
. M , A .
. M . URL
. A .
( , ), .
```

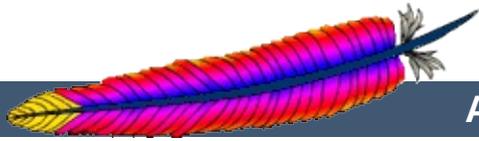
```
:
#
ExpiresActive On
# GIF
ExpiresByType image/gif A2592000
# HTML ExpiresByType text/html M604800
```

```
ExpiresActive On ExpiresDefa
MIME type .
```



## ExpiresDefault

```
: ExpiresDefault <code>seconds  
: , , directory, .htaccess  
Override : Indexes  
: Extension  
: mod_expires
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_ext\_filter

- 
- Extension
- ext\_filter\_module
- mod\_ext\_filter.c

mod\_ext\_filter . (,  
) . API  
, :  
•  
• /  
•  
, mod\_ext\_filter



## 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*

```

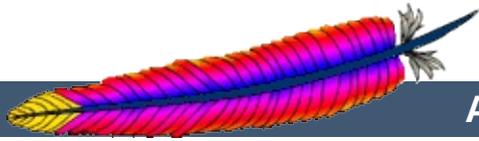
DebugLevel mod_ext_filter .
. DebugLevel=0 . ,
. mod_ext_filter.c DBGLV
.
: core LogLevel .
```

### LogStderr | NoLogStderr

```

LogStderr .
NoLogStderr .
```

```
ExtFilterOptions LogStderr DebugLevel=0
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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()

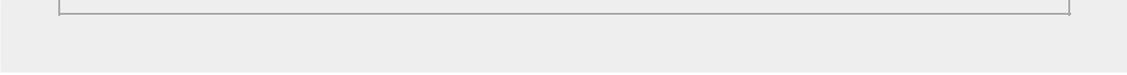
## CacheFile

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
```

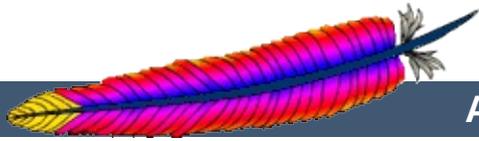


## MMapFile

```
┆  
┆ 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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).

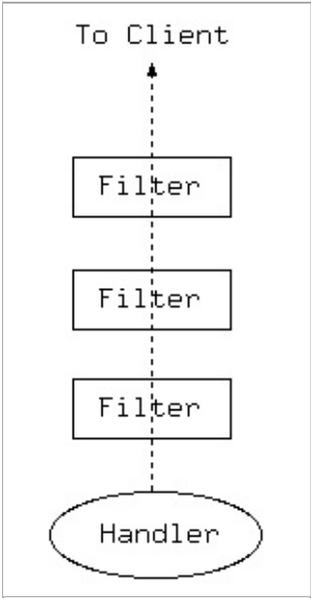
`mod_filter` works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to `mod_filter`; no change to existing filter modules is required (although it may be possible to simplify them).



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.

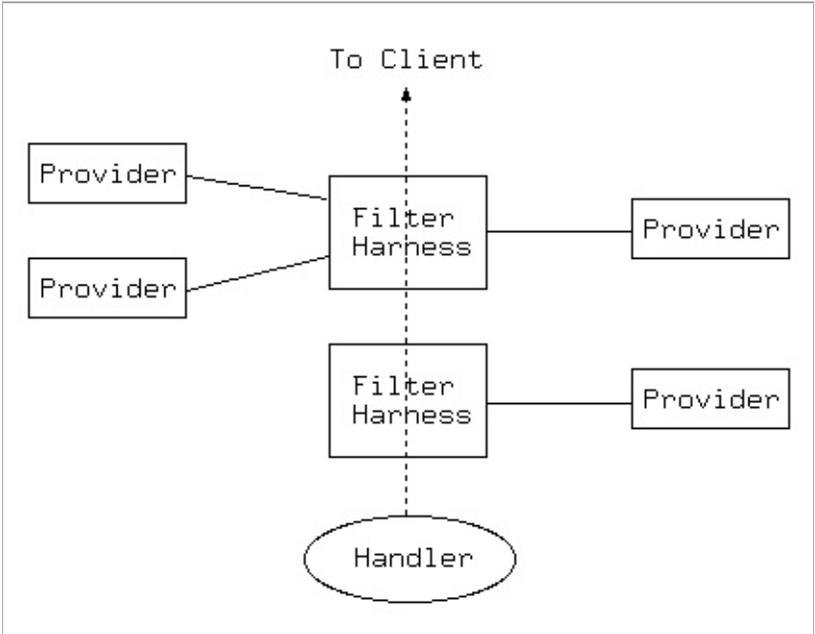
[mod\\_filter](#) by contrast gives server administrators a great deal of flexibility in configuring the filter chain. In fact, filters can be inserted based on any Request Header, Response Header or Environment Variable. This generalises the limited flexibility offered by [AddOutputFilterByType](#), and fixes it to work correctly with dynamic content, regardless of the content generator. The ability to dispatch based on Environment Variables offers the full flexibility of configuration with [mod\\_rewrite](#) to anyone who needs it.





**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 remaining arguments to `FilterProvider` are a dispatch criterion and a match string. The former may be an HTTP request or response header, an environment variable, or the Handler used by this request. The latter is matched to it for each request, to determine whether this provider will be used to implement the filter for this request.

### 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`.



## Server side Includes (SSI)

A simple case of using `mod_filter` in place of `AddOutputFilterByType`

```
FilterDeclare SSI
FilterProvider SSI INCLUDES resp=Content-Type $text/html
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 server-parsed
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 `CONTENT_SET`.

```
FilterDeclare gzip CONTENT_SET
FilterProvider gzip inflate req=Accept-Encoding !$gzip
FilterChain gzip
```

## Image Downsampling

Suppose we want to downsample all web images, and have filters for GIF, JPEG and PNG.

```
FilterProvider unpack jpeg_unpack Content-Type $image/jpeg
FilterProvider unpack gif_unpack Content-Type $image/gif
FilterProvider unpack png_unpack Content-Type $image/png

FilterProvider downsample downsample_filter Content-Type
$image
FilterProtocol downsample "change=yes"

FilterProvider repack jpeg_pack Content-Type $image/jpeg
FilterProvider repack gif_pack Content-Type $image/gif
FilterProvider repack png_pack Content-Type $image/png
```

```
<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.



<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.

*proto-flags* is one or more of

#### **change=yes**

The filter changes the content, including possibly the content length

#### **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> [req resp env]=<i>dispatch</i> <i>match</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 *match* declared here matches the value of the header or environment variable declared as *dispatch*.

*provider-name* must have been registered by loading a module that registers the name with `ap_register_output_filter`.

The *dispatch* argument is a string with optional `req=`, `resp=` or `env=` prefix causing it to dispatch on (respectively) the request header, response header, or environment variable named. In the absence of a prefix, it defaults to a response header. A special case is the word `handler`, which causes `mod_filter` to dispatch on the content handler.

The *match* argument specifies a match that will be applied to the filter's *dispatch* criterion. The *match* may be a string match (exact match or substring), a [regex](#), an integer (greater, less than or equals), or unconditional. The first characters of the *match* argument determines this:

**First**, if the first character is an exclamation mark (!), this reverses the rule, so the provider will be used if and only if the match *fails*.

**Second**, it interprets the first character excluding any leading ! as follows:

Character	Description
<i>(none)</i>	exact match
\$	substring match
/	regex match (delimited by a second /)
=	integer equality
<	integer less-than
<=	integer less-than or equal
>	integer greater-than
>=	integer greater-than or equal
*	Unconditional match



<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)**

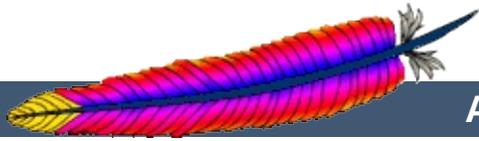
No debug information is generated.

#### **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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_headers

[: HTTP](#)

[: Extension](#)

[: headers\\_module](#)

[: mod\\_headers.c](#)

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 .

```



## 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 .
```

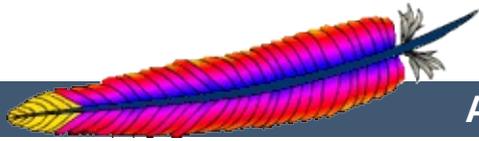
fixup

Requesti

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_ident

[RFC 1413 ident](#)

[Extension](#)

[ident\\_module](#)

[mod\\_ident.c](#)

[2.1](#)

[RFC 1413](#) .

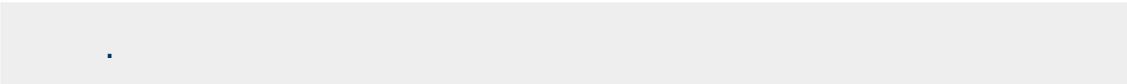
[mod\\_log\\_config](#)



## Identity Check

- [RFC 1413](#)
- [IdentityCheck On|Off](#)
- [IdentityCheck Off](#)
- [, , directory](#)
- [Extension](#)
- [mod\\_ident](#)
- [2.1 core](#)

[RFC 1413](#) identd  
%...1 .



[IdentityCheckTimeout](#)

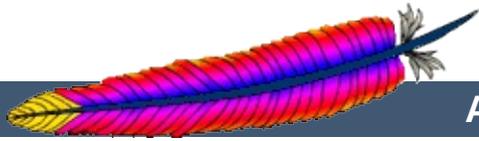


## IdentityCheckTimeout

```
:_ ident
:_ IdentityCheckTimeout seconds
:_ IdentityCheckTimeout 30
:_ , , directory
:_ Extension
:_ mod_ident
```

ident .

[RFC](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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-imap 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

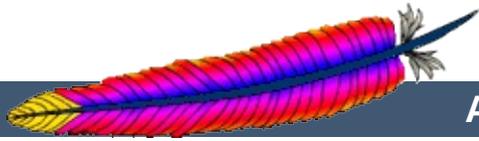
formatted .  
 . , .

### semiformatted

semiformatted . HTML .  
 , formatted .

### unformatted

, . .  
 . , .  
 .



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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
<b>Compatibility:</b>	Implemented as an output filter since Apache 2.0

## 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.

## See also

[Options](#)

[AcceptPathInfo](#)

[Filters](#)

[SSI Tutorial](#)



## Enabling Server Side Includes

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="#">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 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.

### **errmsg**

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.

### **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".

### **timefmt**

The value is a string to be used by the `strftime(3)` library routine when printing dates.

## **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.

### **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. 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, and only special characters as defined in the ISO-8859-1 character encoding will be encoded. This encoding process may not have the desired result if a different character encoding is in use.

In order to avoid cross-site scripting issues, you should *always* encode user supplied data.

## 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.

### **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.

## The `flastmod` 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; the inclusion is done for each attribute given to the include command. 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.

## The printenv Element

This prints out a 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

```
<!--#printenv -->
```

## 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.

### 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\_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, this variable contains the (%-decoded) query string, which is *escaped* for shell usage (special characters like `&` etc. are preceded by backslashes).



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, as well as the arguments to conditional operators. You can insert a literal dollar sign into the string using backslash quoting:

```
<!--#if expr="$a = \$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 below example will print "in foo" if the DOCUMENT\_URI is /foo/file.html, "in bar" if it is /bar/file.html and "in neither" otherwise:

```
<!--#if expr="$DOCUMENT_URI" = "/foo/file.html" -->
  in foo
<!--#elif expr="$DOCUMENT_URI" = "/bar/file.html" -->
  in bar
<!--#else -->
  in neither
<!--#endif -->
```



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 one of the following:

### ***string***

true if *string* is not empty

### ***-A string***

true if the URL represented by the string is accessible by configuration, false otherwise. This test only has an effect if **SSIEnableAccess** is on. 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`.

### 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.



**Description:** Enable the -A flag during conditional flow control processing.

**Syntax:** SSIEnableAccess on|off

**Default:** SSIEnableAccess off

**Context:** directory, .htaccess

**Status:** Base

**Module:** mod\_include

The `SSIEnableAccess` directive controls whether the -A test is enabled during conditional flow control processing.

`SSIEnableAccess` can take on the following values:

#### **off**

`<!--#if expr="-A /foo"-->` will be interpreted as a series of string and regular expression tokens, the -A has no special meaning.

#### **on**

`<!--#if expr="-A /foo"-->` will evaluate to false if the URL /foo is inaccessible by configuration, or true otherwise.



<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
<b>Compatibility:</b>	Available in version 2.0.30 and later.

This directive changes the string that [mod\\_include](#) looks for to mark the end of an include element.

### Example

```
SSIEndTag "%>"
```

### See also

- [SSIStartTag](#)



<b>Description:</b>	Error message displayed when there is an SSI error
<b>Syntax:</b>	SSIErrorMsg <i>message</i>
<b>Default:</b>	SSIErrorMsg "[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
<b>Compatibility:</b>	Available in version 2.0.30 and later.

The `SSIErrorMsg` 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.

**Example**

```
SSIErrorMsg "<!-- 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`.



<b>Description:</b>	String that starts an include element
<b>Syntax:</b>	<code>SSIStartTag tag</code>
<b>Default:</b>	<code>SSIStartTag "&lt;! - -#"</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_include</code>
<b>Compatibility:</b>	Available in version 2.0.30 and later.

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).

### Example

```
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
<b>Compatibility:</b>	Available in version 2.0.30 and later.

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.

### Example

```
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
<b>Compatibility:</b>	Available in version 2.0.34 and later.

This directive changes the string that `mod_include` displays when a variable is not set and "echoed".

### Example

```
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**

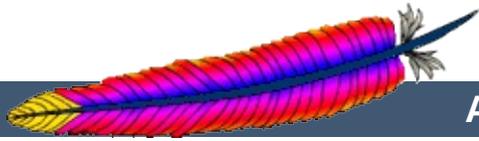
You would not want to use the full option, unless you assure the group-execute bit is unset for every SSI script which might `#include` a CGI or otherwise produces different output on each hit (or could potentially change on subsequent requests).

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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

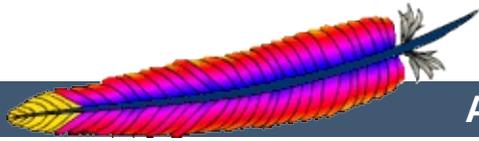
- [LoadModule](#), [LoadFile](#)
- [Include](#), [<IfModule>](#), [<IfDefine>](#)
- [<Directory>](#) , [mod\\_info](#)
- [mod\\_perl](#)



```
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/2.2/mod/mod_deflate.html">\
http://www.apache.org/docs/2.2/mod/mod_deflate.html</a>'
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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
<b>Override :</b>	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
<a href="#">Override :</a>	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

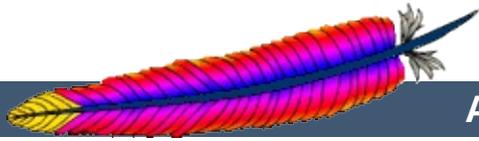
```
ISAPI extension
ISAPILogNotSupported on|off
ISAPILogNotSupported off
, , directory, .htaccess
Override : FileInfo
Base
mod_isapi
```

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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
<b>Compatibility:</b>	Available in version 2.0.41 and later

### 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), native Microsoft LDAP SDK, or the [iPlanet \(Netscape\) 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_authnz_ldap
# be loaded. Change the "yourdomain.example.com" to
# match your domain.

LDAPSharedCacheSize 500000
LDAPCacheEntries 1024
LDAPCacheTTL 600
LDAPOpCacheEntries 1024
LDAPOpCacheTTL 600

<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one
  AuthzLDAPAuthoritative off
  Require valid-user
</Location>
```



---

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.



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.

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. Requires that
# mod_ldap and mod_authnz_ldap be loaded. Change the
# "yourdomain.example.com" to match your domain.
```

```
LDAPTrustedGlobalCert CA_DER /certs/certfile.der
```

```
<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
  AuthzLDAPAuthoritative off
  Require valid-user
</Location>
```

```
# Establish a TLS LDAP connection on port 389. Requires that
# mod_ldap and mod_authnz_ldap be loaded. Change the
# "yourdomain.example.com" to match your domain.
```

```
LDAPTrustedGlobalCert CA_DER /certs/certfile.der
```

```
<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one TLS
  AuthzLDAPAuthoritative off
  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/cert7.db
# Specify an optional key3.db file for client certificate
support
LDAPTrustedGlobalCert CERT_KEY3_DB /certs/key3.db
# Specify the secmod file if required
LDAPTrustedGlobalCert CA_SECMOD /certs/secmod
<Location /ldap-status>
    SetHandler ldap-status
    Order deny,allow
    Deny from all
```

```
Allow from yourdomain.example.com
LDAPTrustedClientCert CERT_NICKNAME <nickname> [password]
AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
AuthzLDAPAuthoritative off
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/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
# Specify a client certificate file and key
LDAPTrustedGlobalCert CERT_BASE64 /certs/cert1.pem
LDAPTrustedGlobalCert KEY_BASE64 /certs/key1.pem [password]
# Do not use this directive, as it will throw an error
#LDAPTrustedClientCert CERT_BASE64 /certs/cert1.pem
```

## 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.

Client certificates are specified per connection using the LDAPTrustedClientCert directive.

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/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  LDAPTrustedClientCert CERT_BASE64 /certs/cert1.pem
  LDAPTrustedClientCert KEY_BASE64 /certs/key1.pem
  AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
  AuthzLDAPAuthoritative off
  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>	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` 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` option, and the ultimate behavior is dictated entirely by the LDAP client library.



**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.



<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.



<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>	server config, virtual host, 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:

- 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>	<code>LDAPTrustedGlobalCert <i>type directory-path/filename [password]</i></code>
<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.



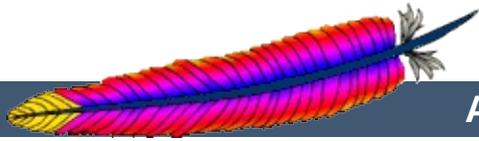
<b>Description:</b>	Force server certificate verification
<b>Syntax:</b>	LDAPVerifyServerCert <i>On Off</i>
<b>Default:</b>	LDAPVerifyServerCert On
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ldap

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_log\_config

- 
- 
- Base
- log\_config\_module
- mod\_log\_config.c

· TransferLog , LogFormat ,  
CustomLog . TransferLog CustomL



## LogFormat CustomLog

C "\n" "\t"

" %" .

%%	
%. . . a	IP-
%. . . A	() IP-
%. . . B	HTTP .
%. . . b	HTTP . CLF 0 ' -' .
%. . . {Foobar}C	<i>Foobar</i> .
%. . . D	().
%. . . {FOOBAR}e	FOOBAR
%. . . f	
%. . . h	
%. . . H	
%. . . {Foobar}i	<i>Foobar:</i> .
%. . . l	( identd ) . <u>mod_ident</u> <u>IdentityCheck On</u> .
%. . . m	
%. . . {Foobar}n	<i>Foobar (note)</i> .
%. . . {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

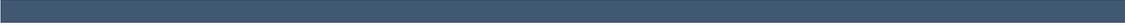
```
"%{Referer}i -> %U"
```

### Agent ()

```
"%{User-agent}i"
```

```
ServerName Listen      %v %p .
```





- [\[:\]](#) Buffer log entries in memory before writing to disk
- [\[:\]](#) BufferedLogs On|Off
- [\[:\]](#) BufferedLogs Off
- [\[:\]](#)
- [\[:\]](#) Base
- [\[:\]](#) mod\_log\_config
- [\[:\]](#) Available in versions 2.0.41 and later.

The documentation for this directive has not been translated yet. Please have a look at the English version.



```
CookieLog filename  
Base  
mod_log_config  
.
```

```
CookieLog  
mod_cookies ,
```



## 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
```



## 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
```

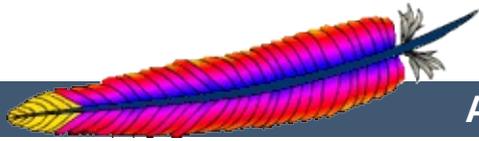


## TransferLog

```
:\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
```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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>	<code>log_forensic_module</code>
<b>Source File:</b>	<code>mod_log_forensic.c</code>
<b>Compatibility:</b>	<code>mod_unique_id</code> 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.

## 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.

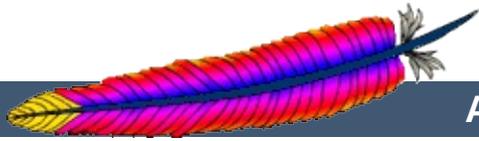
### **Note**

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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_logio

- 
- 
- Extension
- logio\_module
- mod\_logio.c

SSL/TLS, SSL/TLS

mod\_log\_config.

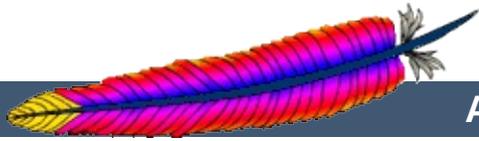
mod\_log\_config



. " %" : :

%. . . I	. 0
%. . . 0	. 0

: : "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\" %I %0"



| [FAQ](#) |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_mem\_cache

- URI
- Experimental
- mem\_cache\_module
- mod\_mem\_cache.c

...

```
mod_cache . mod_cache .  
mod mem cache .  
mod mem cache ProxyPass ( (   
mod proxy .
```

URI .

```
mod_cache  
mod_disk_cache
```



## MCacheMaxObjectCount

```
MCacheMaxObjectCount value
MCacheMaxObjectCount 1009
Experimental
mod_mem_cache
```

MCacheMaxObjectCount .

MCacheRemovalAlgorithm .

MCacheMaxObjectCount 13001



## MCacheMaxObjectSize

```
MCacheMaxObjectSize ()  
MCacheMaxObjectSize bytes  
MCacheMaxObjectSize 10000  
Experimental  
mod_mem_cache
```

MCacheMaxObjectSize .

```
MCacheMaxObjectSize 6400000
```

**Note**  
MCacheMaxObjectSize MCacheMinObjectSize  
.



## MCacheMaxStreamingBuffer

```

:
: MCacheMaxStreamingBuffer size_in_bytes
: MCacheMaxStreamingBuffer 100000
  MCacheMaxObjectSize
:
: Experimental
: mod_mem_cache

```

### MCacheMaxStreamingBuffer

. (streamed response)  
Content-Length . CGI  
Content-Length .  
Content-Length . MCacheMaxStre

```

:
: MCacheMaxStreamingBuffer 0
. mod_mem_cache

```

```

# 64KB :
MCacheMaxStreamingBuffer 65536

```



## MCacheMinObjectSize

```
┆  ()  
┆ MCacheMinObjectSize bytes  
┆ MCacheMinObjectSize 0  
┆  
┆ Experimental  
┆ mod_mem_cache
```

### MCacheMinObjectSize .

```
MCacheMinObjectSize 10000
```



## MCacheRemovalAlgorithm

- MCacheRemovalAlgorithm LRU|GDSF
- MCacheRemovalAlgorithm GDSF
- Experimental
- mod\_mem\_cache

MCacheRemovalAlgorithm .

### LRU (Least Recently Used)

LRU .

### GDSF (GreedyDual-Size)

GDSF (cache miss) .

MCacheRemovalAlgorithm GDSF  
MCacheRemovalAlgorithm LRU



```

: (KByte
)
: MCacheSize KBytes
: MCacheSize 100
:
: Experimental
: mod_mem_cache

```

**MCacheSize** KByte (1024 ) .

MCacheRemovalAlgorithm .

```

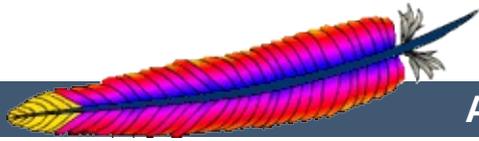
MCacheSize 700000

```

```

MCacheSize MCacheMaxObjectSize .

```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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>	mime_module
<b>Source File:</b>	mod_mime.c

## Summary

This module is used to associate various bits of "meta information" with files by their filename extensions. This information relates the filename of the document to its mime-type, language, character set and encoding. This information is sent to the browser, and participates in content negotiation, so the user's preferences are respected when choosing one of several possible files 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 meta-information for that file. Respectively they set the character set, content-encoding, content-language, and [MIME-type](#) (content-type) of documents. The directive [TypesConfig](#) is used to specify a file which also maps extensions onto MIME 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 meta-information 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 meta-information. These directives include [ForceType](#), [SetHandler](#), [SetInputFilter](#), and [SetOutputFilter](#). The core directives override any filename extension mappings defined in [mod\\_mime](#).

Note that changing the meta-information 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 meta-information (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.

## See also

- [MimeMagicFile](#)
- [AddDefaultCharset](#)
- [ForceType](#)
- [DefaultType](#)
- [SetHandler](#)
- [SetInputFilter](#)
- [SetOutputFilter](#)



## FILES WITH MULTIPLE EXTENSIONS

Files can have more than one extension, and 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 meta-information, then the one to the right will be used, except for languages and content encodings. For example, if `.gif` maps to the [MIME-type](#) `image/gif` and `.html` maps to the MIME-type `text/html`, then the file `welcome.gif.html` will be associated with the MIME-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 [MIME-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 MIME-type `text/html`, then the file `world.imap.html` will be associated with both the `imap-file` handler and `text/html` MIME-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 \.cgi$>  
  SetHandler cgi-script  
</FilesMatch>
```



A file of a particular [MIME-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 <i>charset</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 **AddCharset** directive maps the given filename extensions to the specified content charset. *charset* is the [MIME charset parameter](#) of 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>MIME-enc 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 encoding type. *MIME-enc* is the MIME encoding to use for documents containing 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
<b>Compatibility:</b>	AddInputFilter is only available in Apache 2.0.26 and later.

`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>MIME-lang 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. *MIME-lang* is the MIME language of filenames containing *extension*. This mapping is added to any already in force, overriding 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 Directives

<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
<b>Compatibility:</b>	AddOutputFilter is only available in Apache 2.0.26 and later.

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 `.shtml` files for server-side includes and will then compress the output using `mod_deflate`.

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

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.

## See also

- [RemoveOutputFilter](#)
- [SetOutputFilter](#)



## AddType Directive

**Description:** Maps the given filename extensions onto the specified content type

**Syntax:** `AddType MIME-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. *MIME-type* is the [MIME 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*. This directive can be used to add mappings not listed in the MIME types file (see the [TypesConfig](#) directive).

### Example

```
AddType image/gif .gif
```

It is recommended that new MIME types be added using the `AddType` directive rather than changing the [TypesConfig](#) file.

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

- [DefaultType](#)
- [ForceType](#)



## DefaultLanguage Directive

<b>Description:</b>	Sets all files in the given scope to the specified language
<b>Syntax:</b>	DefaultLanguage <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_mime

The **DefaultLanguage** directive tells Apache that all files in the directive's scope (e.g., all files 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 considered to be in the specified *MIME-lang* language. This allows entire directories 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 that file will be considered to have no language attribute.

### 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>
<b>Compatibility:</b>	Available in Apache 2.0.41 and later

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 `/bar/foo.shtml` where `/bar` is a Location and `ModMimeUsePathInfo` is `On`, `mod_mime` will treat the incoming request as `/bar/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.

### 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
<b>Compatibility:</b>	Available in Apache 2.0.26 and later.

**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 was the behavior in Apache 1.3, and 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
<b>Compatibility:</b>	RemoveCharset is only available in Apache 2.0.24 and later.

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
<b>Compatibility:</b>	RemoveInputFilter is only available in Apache 2.0.26 and later.

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
<b>Compatibility:</b>	RemoveLanguage is only available in Apache 2.0.24 and later.

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.



<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
<b>Compatibility:</b>	RemoveOutputFilter is only available in Apache 2.0.26 and later.

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](#)



## RemoveType Directive

<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 **MIME 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 the files to be treated as being of the **DefaultType**.

### **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 [MIME 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 provided `mime.types` file, which associates common filename extensions with IANA registered content types. The current list is maintained at <http://www.iana.org/assignments/media-types/index.html>. 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:

```
MIME-type [extension] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (`#`) are ignored.

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.

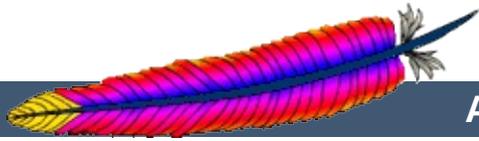
## See also

- [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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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  \<MIFFfile   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.

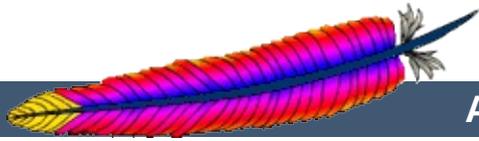


<b>Description:</b>	Enable MIME-type determination based on file contents using the specified magic file
<b>Syntax:</b>	MimeMagicFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_mime_magic

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

## 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 :**

New in Apache 2.0, 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-----
```



## MULTI-VIEWS

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
<b>Compatibility:</b>	The syntax changed in version 2.0.

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.

Prior to version 2.0, **CacheNegotiatedDocs** did not take an argument; it was turned on by the presence of the directive by itself.



## 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
<b>Compatibility:</b>	Available in version 2.0.30 and later

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.

### Example:

```
LanguagePriority en fr de
```

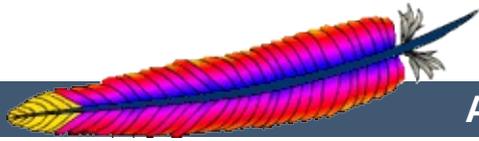
For a request for `foo.html`, where `foo.html.fr` and `foo.html.de` both existed, but the browser did not express a language preference, then `foo.html.fr` would be returned.

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](#)



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.



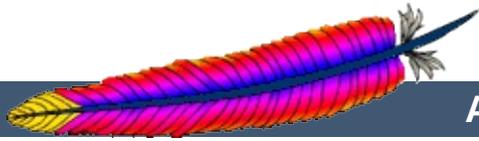
<b>Description:</b>	Enables SSL encryption for the specified port
<b>Syntax:</b>	<code>SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

# Apache Module `mod_proxy`

<b>Description:</b>	HTTP/1.1 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.

This module implements a proxy/gateway for Apache. It implements proxying capability for AJP13 (Apache JServe Protocol version 1.3), FTP, CONNECT (for SSL), HTTP/0.9, HTTP/1.0, and HTTP/1.1. The module can be configured to connect to other proxy modules for these and other protocols.

Apache's proxy features are divided into several modules in addition to `mod_proxy`: `mod_proxy_http`, `mod_proxy_ftp`, `mod_proxy_ajp`, `mod_proxy_balancer`, and `mod_proxy_connect`. Thus, if you want to use one or more of the particular proxy functions, load `mod_proxy` and the appropriate module(s) into the server (either statically at compile-time or dynamically via the [LoadModule](#) directive).

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.

## See also

[mod\\_cache](#)

[mod\\_proxy\\_http](#)

[mod\\_proxy\\_ftp](#)

[mod\\_proxy\\_connect](#)

[mod\\_proxy\\_balancer](#)

[mod\\_ssl](#)



---

Apache 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 *>
  Order deny,allow
  Deny from all
  Allow from internal.example.com
</Proxy>
```



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 pooling. 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
connectiontimeout=5 timeout=30
```

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/examples  
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  
timeout=10
```

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 and some configuration defaults will overwrite the configuration given for the first worker. This will be logged as a warning. In the

above example, the resulting timeout value for the URL /apps will be 10 instead of 60!

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`, `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](#).



You can control who can access your proxy via the [<Proxy>](#) control block as in the following example:

```
<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from 192.168.0
</Proxy>
```

For more information on access control directives, see [mod\\_auth\\_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.



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 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 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.



## HTTP Requirements

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>
```



## 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.



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.



<b>Description:</b>	Ports that are allowed to CONNECT through the proxy
<b>Syntax:</b>	AllowCONNECT <i>port</i> [ <i>port</i> ] ...
<b>Default:</b>	AllowCONNECT 443 563
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `AllowCONNECT` directive specifies a list of port numbers 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 `snews` port (563) are enabled. Use the `AllowCONNECT` directive to override this default and allow connections to the listed ports only.

Note that you'll need to have `mod_proxy_connect` present in the server in order to get the support for the CONNECT at all.



<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 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 pairs available to [ProxyPass](#) directives.

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.



<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 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 .apache.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.com  
www.apache.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 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 *>
  Order Deny,Allow
  Deny from all
  Allow from 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>](#)



## 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
<b>Compatibility:</b>	Available in Apache 2.0.44 and later

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 joes-garage.com some-host.co.uk  
rocky.wotsamattau.edu
```

rocky.wotsamattau.edu would also be matched if referenced by IP address.

Note that wotsamattau would also be sufficient to match wotsamattau.edu.

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 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
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in version 2.0 and later

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.



## 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
<b>Compatibility:</b>	Available in Apache 2.2.7 and later

The `ProxyFtpDirCharset` directive defines the character set to be set for FTP directory listings in HTML generated by [mod\\_proxy\\_ftp](#).



<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 8192.

When the `mod_proxy_ajp` module is used, this value is aligned to a 1024 byte boundary, and values larger than 65536 are set to 65536 in accordance with the AJP protocol.

In almost every case, there's no reason to change that value.



## ProxyMatch Directive

<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](#).

### 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>	Available in Apache 2.0 and later; 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 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>	<code>ProxyPass [path] ! url [key=value [key=value ...]] [nocanon] [interpolate]</code>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

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.

The `ProxyRequests` directive should usually be set **off** when using `ProxyPass`.

Suppose the local server has address `http://example.com/`; then

```
ProxyPass /mirror/foo/ http://backend.example.com/
```

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 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.

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.

The ProxyPass directive is not supported in `<Directory>` or `<Files>` sections.

If you require a more flexible reverse-proxy configuration, see the `RewriteRule` directive with the [P] flag.

The ! directive is useful in situations where you don't want to reverse-proxy a subdirectory, e.g.

```
ProxyPass /mirror/foo/i !  
ProxyPass /mirror/foo http://backend.example.com
```

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.

For the same reasons, exclusions must come *before* the general `ProxyPass` directives.

### Ordering ProxyPass and RewriteRule Directives

`RewriteRule` directives are evaluated before `ProxyPass` ones.

### 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.

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. 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.

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.

#### Example

```
ProxyPass /example http://backend.example.com max=20 ttl=120  
retry=300
```

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 <code>ThreadsPerChild</code> 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 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.
acquire	-	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 will return SERVER_BUSY status to the client.
connectiontimeout	timeout	Connect timeout in seconds. The number of seconds Apache 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.
disablereuse	Off	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 persistent connection and pool for that backend. This helps in various situations where a firewall between

		<p>Apache and the backend server (regardless of protocol) tends to silently drop connections or when backends themselves may be under round- robin DNS. To disable connection pooling reuse, set this property value to On.</p>
flushpackets	off	<p>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.</p>
flushwait	10	<p>The time to wait for additional input, in milliseconds, before flushing the output brigade if 'flushpackets' is 'auto'.</p>
keepalive	Off	<p>This parameter should be used when you have a firewall between your Apache and the backend server, which tends to drop inactive connections. This</p>

flag will tell the Operating System to send KEEP\_ALIVE 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 webserver to send a CPING request on ajp13 connection before forwarding a request. The parameter is the delay in seconds to wait for the CPONG reply. This features

has been added to avoid problem with hung and busy Tomcat's and require ajp13 ping/pong support which has been implemented on Tomcat 3.3.2+, 4.1.28+ and 5.0.13+. 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. Currently, this has an effect only for AJP. By adding a postfix of ms, the delay can be also set in milliseconds.

loadfactor            1

Worker load factor. Used with BalancerMember. It is a number between 1 and 100 and defines the normalized weighted load applied to the worker.

redirect              -

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.

retry	60	Connection pool worker retry timeout in seconds. If the connection pool worker to the backend server is in the error state, Apache will not 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' is disabled; 'S' is stopped; 'I' is ignore-errors; 'H' is hot-standby; and 'E' is in an error state. 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	<u>ProxyTimeout</u>	Connection timeout in seconds. The number of seconds Apache waits for

ttl	-	<p>data sent by / to the backend.</p> <p>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.</p>
-----	---	---

If the **ProxyPass** 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.

Parameter	Default	Description
lbmethod	byrequests	Balancer load-balance method. Select the load-balancing scheduler method use. Either byrequests, to perform weighted request counting; bytraffic to perform weighted traffic byte count balancing; or bybusyness (Apache HTTP Server 2.2.10 and later), to perform pending request balancing. The default is byrequests.
maxattempts	One less than the number of workers,	Maximum number of failover attempts before giving up.

		or 1 with a single worker.
nofailover	Off	If set to On, the session will break if the worker is in error state or disabled. Set this value to On if backend servers do not support session replication.
stickysession	-	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 different names for cookies and url encoded id (like servlet containers), use   to separate them. The first part is the cookie; the second for the path.
scolonpathdelim	Off	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 JSESSIONID=6736bcf34;foo=aa
timeout	0	Balancer timeout in seconds. If set, this will be the maximum time to wait for a free worker. The default is to not wait.
failonstatus	-	A single or comma-separated list of 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. Available with

failontimeout	Off	Apache HTTP Server 2.2.17 and later 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. Available with Apache HTTP Server 2.2.25 and later.
forcerecovery	On	Force the immediate recovery of all workers without considering the retry parameter of the workers if all workers of a balancer are in error state. There might be cases where an already overloaded backend can get into deep trouble if the recovery of all workers is enforced without considering the retry parameter of each worker. In this case set to Off. Available with Apache HTTP Server 2.2.23 and later.

### A sample balancer setup:

```
ProxyPass /special-area http://special.example.com smax=5
max=10
ProxyPass / balancer://mycluster/
stickysession=JSESSIONID|jsessionid nofailover=0n
<Proxy balancer://mycluster>
  BalancerMember ajp://1.2.3.4:8009
  BalancerMember ajp://1.2.3.5:8009 loadfactor=20
  # Less powerful server, don't send as many requests there,
  BalancerMember ajp://1.2.3.6:8009 loadfactor=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=2
# The below is the hot standby
BalancerMember ajp://1.2.3.6:8009 status=+H
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.

The optional `interpolate` keyword (available in `httpd 2.2.9` and later), in combination with `ProxyPassInterpolateEnv`, causes the ProxyPass to interpolate environment variables, using the syntax `#{VARNAME}`. 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}
ProxyPassReverse /mirror/foo/ http://backer
```

```
ProxyPassReverse /mirror/foo/ https://backe
```



## 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
<b>Compatibility:</b>	available in Apache 2.2.5 and later

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*.

Suppose the local server has address `http://example.com/`; then

```
ProxyPassMatch ^(/.*\.gif)$ http://backend.example.com$1
```

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:8000$1
```

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:8000/$1
```

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.

### **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 adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache 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 will not rewrite other response headers, nor will it rewrite URL references inside HTML pages. This means that if the proxied content contains absolute URL references, they will bypass the proxy. A third-party module that will look inside the HTML and rewrite URL references is Nick Kew's [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://backend.example.com/  
ProxyPassReverse /mirror/foo/ http://backend.example.com/  
ProxyPassReverseCookieDomain backend.example.com  
public.example.com  
ProxyPassReverseCookiePath / /mirror/foo/
```

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 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 (available in httpd 2.2.9 and later), 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 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/foo/
```

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
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.31 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 module="mod_proxy"` 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.example.com:8000  
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 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 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 timeout=15
```

```
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.



## 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
<b>Compatibility:</b>	Available in Apache 2.0.31 and later

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>

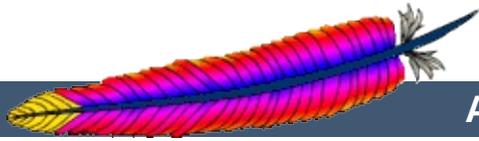
This directive controls the use of the `Via`: HTTP header by the proxy. Its intended use is to control the flow of proxy requests along a chain of proxy servers. See [RFC 2616](#) (HTTP/1.1), section 14.45 for an explanation of `Via`: header lines.

- 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 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

## 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

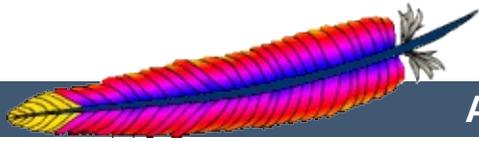
The container asks for more data from the request (If the body was too large to fit in the first packet sent over or when the request is chunked). The server will send a body packet back with an amount of data which is the minimum of the request\_length, the maximum send body size (8186 (8 Kbytes - 6)), 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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`. It provides load balancing support for HTTP, FTP and AJP13 protocols

Thus, in order to get the ability of load balancing, `mod_proxy` and `mod_proxy_balancer` 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.

## See also

[mod\\_proxy](#)



## Load balancer scheduler algorithms

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.



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 a 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
```

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_WORKER_ROUTE}e;
path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://mycluster>
BalancerMember http://192.168.1.50:80 route=1
BalancerMember http://192.168.1.51:80 route=2
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /test balancer://mycluster
```



## 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 worker's 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.



## Weighted Round Robin 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 seen.

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.



## Request Counting Algorithm

Enabled via `lbmethod=bybusyness`, this scheduler keeps track of how many requests each worker is 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 fastest.

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`.

This algorithm is available in Apache HTTP Server 2.2.10 and later.



---

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

  Order Deny,Allow
  Deny from all
  Allow from .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`



## 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](#).

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 `sColonPathDelimiter` 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
stickysession=JSESSIONID|jsessionid sColonPathDelimiter=On
<Proxy balancer://mycluster>
BalancerMember http://192.168.1.50:80 route=node1
BalancerMember http://192.168.1.51:80 route=node2
</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.

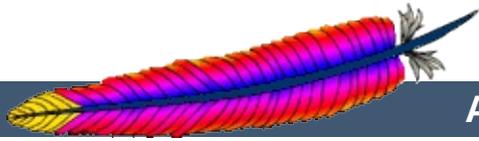
Common reasons for loss of session are session timeouts, which are usually configurable on the back-end server.

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

## See also

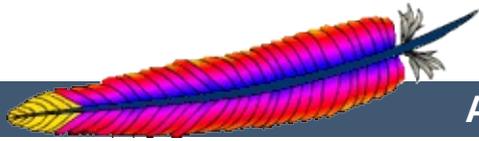
[AllowCONNECT](#)

[mod\\_proxy](#)

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

## 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:

```
DefaultType 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

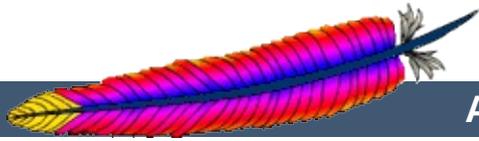
The password which is transmitted in such a way is not encrypted on its way. It travels between your browser and the

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.

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

## 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**

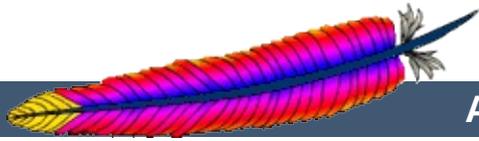
This is the opposite of *proxy-sendcl*. It allows request bodies to be sent to the backend using chunked transfer encoding. This allows the request to be efficiently streamed, but requires that the backend server supports HTTP/1.1.

### **proxy-interim-response**

This variable takes values RFC 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 connection is an initial 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.



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Modules](#)

# Apache Module `mod_proxy_scgi`

<b>Description:</b>	SCGI gateway module for <a href="#">mod_proxy</a>
<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.

## See also

[mod\\_proxy](#)

[mod\\_proxy\\_balancer](#)



## Examples

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](#) in addition to the already mentioned proxy modules.

### Balanced gateway

```
ProxyPass /scgi-bin/ balancer://somecluster/  
<Proxy balancer://somecluster/>  
  BalancerMember scgi://localhost:4000/  
  BalancerMember scgi://localhost:4001/  
</Proxy>
```



<b>Description:</b>	Enable or disable internal redirect responses from the backend
<b>Syntax:</b>	ProxySCGIInternalRedirect On Off
<b>Default:</b>	ProxySCGIInternalRedirect On
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_scgi

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` header and its value starts with a slash (/). This value is interpreted as a new local URL the apache internally redirects to.

`mod_proxy_scgi` does the same as `mod_cgi` in this regard, except that you can turn off the feature.

### Example

```
ProxySCGIInternalRedirect Off
```



<b>Description:</b>	Enable evaluation of <i>X-Sendfile</i> pseudo response header
<b>Syntax:</b>	ProxySCGISendfile On Off  <i>Headername</i>
<b>Default:</b>	ProxySCGISendfile Off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy_scgi

The **ProxySCGISendfile** directive enables the SCGI backend to let files serve directly by the gateway. This is useful performance purposes -- the httpd can use `sendfile` or other optimizations, which are not possible if the file comes over the backend socket.

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 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 the argument is applied as header name.

### **Example**

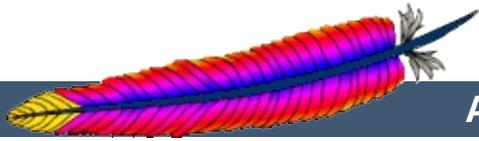
```
# Use the default header (X-Sendfile)
ProxySCGISendfile On
```

```
# 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 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=500
```

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=500  
body=20,MinRate=500
```



<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>	Unset; no limit
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_reqtimeout

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.

- **Timeout value that is increased when data is received:**

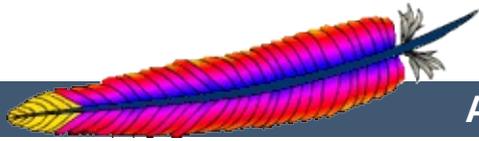
```
type=timeout,MinRate=data_rate
```

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 HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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>
<b>Compatibility:</b>	Available in Apache 1.3 and later

## Summary

This module uses a rule-based rewriting engine (based on a regular-expression parser) to rewrite requested URLs on the fly. 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, of server variables, environment variables, HTTP headers, or time stamps. Even external database lookups in various formats can be used to achieve highly granular URL matching.

This module operates on the full URLs (including the path-info part) both in per-server context (`httpd.conf`) and per-directory context (`.htaccess`) and can generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to an internal proxy throughput.

Further details, discussion, and examples, are provided in the [detailed `mod\_rewrite` documentation](#).

## See also

[Rewrite Flags](#)



## Escaping Special Characters

As of Apache 1.3.20, special characters in *TestString* and *Substitution* strings can be escaped (that is, treated as normal characters without their usual special meaning) by prefixing them with a backslash ('\') character. In other words, you can include an actual dollar-sign character in a *Substitution* string by using '\\$'; this keeps mod\_rewrite from trying to treat it as a backreference.



This module keeps track of two additional (non-standard) CGI/SSI environment variables named `SCRIPT_URL` and `SCRIPT_URI`. These contain the *logical* Web-view to the current resource, while the standard CGI/SSI variables `SCRIPT_NAME` and `SCRIPT_FILENAME` contain the *physical* System-view.

Notice: These variables hold the URI/URL *as they were initially requested*, that is, *before* any rewriting. This is important to note because the rewriting process is primarily used to rewrite logical URLs to physical pathnames.

### Example

```
SCRIPT_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.html
SCRIPT_FILENAME=/u/rse/.www/index.html
SCRIPT_URL=/u/rse/
SCRIPT_URI=http://en1.engelschall.com/u/rse/
```



## Enabling Inheritance

By default, `mod_rewrite` configuration settings from the main server context are not inherited by virtual hosts. To make the main server settings apply to virtual hosts, you must place the following directives in each `<VirtualHost>` section:

```
RewriteEngine On  
RewriteOptions Inherit
```



---

For numerous examples of common, and not-so-common, uses for `mod_rewrite`, see the [extended rewrite documentation](#).



<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 either 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 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
Alias /myapp /opt/myapp-1.2.3
<Directory /opt/myapp-1.2.3>
RewriteEngine On
RewriteBase /myapp/
```

```
RewriteRule ^index\.html$ welcome.html  
</Directory>
```



<b>Description:</b>	Defines a condition under which rewriting will take place
<b>Syntax:</b>	<code>RewriteCond <i>TestString CondPattern</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` ( $0 \leq N \leq 9$ ), which provide access to the grouped parts (in parentheses) of the pattern, from the `RewriteRule` which is subject to the current set of `RewriteCond` conditions..
- **RewriteCond backreferences:** These are backreferences of the form `%N` ( $1 \leq N \leq 9$ ), which provide access to the grouped parts (again, in parentheses) of the pattern, from the last matched `RewriteCond` in the current set of conditions.
- **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:

---

**HTTP headers:**

**connection &**

	<b>request:</b>	
HTTP_USER_AGENT	REMOTE_ADDR	
HTTP_REFERER	REMOTE_HOST	
HTTP_COOKIE	REMOTE_PORT	
HTTP_FORWARDED	REMOTE_USER	
HTTP_HOST	REMOTE_IDENT	
HTTP_PROXY_CONNECTION	REQUEST_METHOD	
HTTP_ACCEPT	SCRIPT_FILENAME	
	PATH_INFO	
	QUERY_STRING	
	AUTH_TYPE	
<b>server internals:</b>	<b>date and time:</b>	<b>special:</b>
DOCUMENT_ROOT	TIME_YEAR	API_VERSION
SERVER_ADMIN	TIME_MON	THE_REQUEST
SERVER_NAME	TIME_DAY	REQUEST_URI
SERVER_ADDR	TIME_HOUR	REQUEST_FILENAME
SERVER_PORT	TIME_MIN	IS_SUBREQ
SERVER_PROTOCOL	TIME_SEC	HTTP_REFERER
SERVER_SOFTWARE	TIME_WDAY	
	TIME	

These variables all correspond to the similarly named HTTP MIME-headers, C variables of the Apache server or struct tm fields of the Unix system. Most are documented 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.

**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.

### **API\_VERSION**

This is the version of the Apache 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 in use (in the release version of Apache 1.3.14, for instance, it is 19990320:10), but is mainly of interest to module authors.

### **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.

### **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.

### **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.

### **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).

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 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 structures and (if not found there) via `getenv()` from the Apache 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*.

**Remember:** *CondPattern* is a *perl compatible regular expression* with some additions:

1. You can prefix the pattern string with a '!' character (exclamation mark) to specify a **non**-matching pattern.
2. There are some special variants of *CondPatterns*. Instead of real regular expression strings you can also use one of the following:
  - '**<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.

- **'-d'** (is **d**irectory)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a directory.
- **'-f'** (is regular **f**ile)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file.
- **'-s'** (is regular file, with **s**ize)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file with size greater than zero.
- **'-l'** (is symbolic **l**ink)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a symbolic link.
- **'-x'** (has **e**xecutable 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.
- **'-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!
- **'-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.

**Note:**

All of these tests can also be prefixed by an exclamation mark (!) to negate their meaning.

3. 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 [OR]
RewriteCond %{REMOTE_HOST} =host2 [OR]
RewriteCond %{REMOTE_HOST} =host3
RewriteRule ...some special stuff for any of these hosts
```

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} ^Mozilla
RewriteRule ^/$ /homepage.max.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx
RewriteRule ^/$ /homepage.min.html [L]

RewriteRule ^/$ /homepage.std.html [L]
```

Explanation: If you use a browser which identifies itself as 'Mozilla' (including Netscape Navigator, Mozilla etc), then you get the max homepage (which could include frames, or other special features). If you use the Lynx browser (which is terminal-based), then you get the min homepage (which could be a version designed for easy, text-only browsing). If neither of these conditions apply (you use any other browser, or your browser identifies itself as something non-standard), you get the std (standard) homepage.



## 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 the module instead of 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>	Sets the name of the lock file used for <a href="#">RewriteMap</a> synchronization
<b>Syntax:</b>	RewriteLock <i>file-path</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

This directive sets the filename for a synchronization lockfile which mod\_rewrite needs to communicate with [RewriteMap programs](#). Set this lockfile to a local path (not on a NFS-mounted device) when you want to use a rewriting map-program. It is not required for other types of rewriting maps.



## RewriteLog Directive

<b>Description:</b>	Sets the name of the file used for logging rewrite engine processing
<b>Syntax:</b>	RewriteLog <i>file-path pipe</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The **RewriteLog** directive sets the name of the file to which the server logs any rewriting actions it performs. If the name does not begin with a slash ('/') then it is assumed to be relative to the *Server Root*. The directive should occur only once per server config.

To disable the logging of rewriting actions it is not recommended to set *Filename* to `/dev/null`, because although the rewriting engine does not then output to a logfile it still creates the logfile output internally. **This will slow down the server with no advantage to the administrator!** To disable logging either remove or comment out the **RewriteLog** directive or use `RewriteLogLevel 0!`

The **RewriteLog** log file format is as follows:

Description	Example
Remote host IP address	192.168.200.166
Remote login name	Will usually be "-"
HTTP user auth name	Username, or "-" if no auth
Date and time of request	[28/Aug/2009:13:09:09 --0400]
Virtualhost and virtualhost ID	[www.example.com/sid#84a650]
Request ID, and whether it's a subrequest	[rid#9f0e58/subreq]

Log entry severity level	(2)
Text error message	forcing proxy-throughput with http://127.0.0.1:8080/index.html

## Security

See the [Apache Security Tips](#) document for details on how your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.

## Example

```
# Log to a file:  
RewriteLog "/usr/local/var/apache/logs/rewrite.log"  
  
# Log to a pipe:  
RewriteLog "|/path/to/parser.pl"
```



<b>Description:</b>	Sets the verbosity of the log file used by the rewrite engine
<b>Syntax:</b>	<code>RewriteLogLevel <i>Level</i></code>
<b>Default:</b>	<code>RewriteLogLevel 0</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_rewrite</code>

The `RewriteLogLevel` directive sets the verbosity level of the rewriting logfile. The default level 0 means no logging, while 9 or more means that practically all actions are logged.

To disable the logging of rewriting actions simply set *Level* to 0. This disables all rewrite action logs.

Using a high value for *Level* will slow down your Apache server dramatically! Use the rewriting logfile at a *Level* greater than 2 only for debugging!

### Example

```
RewriteLogLevel 3
```



<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
<b>Compatibility:</b>	The choice of different dbm types is available in Apache 2.0.41 and later

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.txt
```

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:

- **Standard Plain Text**

MapType: txt, MapSource: Unix filesystem path to valid regular file

This is the standard rewriting map feature where the *MapSource* is a plain ASCII file containing either blank lines, comment lines (starting with a '#' character) or pairs like the following - one per line.

### ***MatchingKey SubstValue***

#### **Example**

```
##  
## map.txt -- rewriting map  
##  
  
Ralf.S.Engelschall    rse    # Bastard Operator From Hell  
Mr.Joe.Average       joe    # Mr. Average
```

```
RewriteMap real-to-user txt:/path/to/file/map.txt
```

- **Randomized Plain Text**

MapType: rnd, MapSource: Unix filesystem path to valid regular file

This is identical to the Standard Plain Text variant above but with a special post-processing feature: After looking up a value it is parsed according to contained ``|" characters

which have the meaning of ``or". In other words they indicate a set of alternatives from which the actual returned value is chosen randomly. For example, you might 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.

Example:

### 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://${servers:static}/$1 [NC,P,L]  
RewriteRule ^/(.*) http://${servers:dynamic}/$1 [P,L]
```

- **Hash File**

MapType: dbm [=type], MapSource: Unix filesystem path to valid regular file

Here the source is a binary format DBM file containing the same contents as a *Plain Text* format file, but in a special representation which is optimized for really fast lookups. The *type* can be sdbm, gdbm, ndbm, or db depending on [compile-time settings](#). If the *type* is omitted, the compile-time default will be chosen.

To create a dbm file from a source text file, use the [httxt2dbm](#) utility.

```
$ htxt2dbm -i mapfile.txt -o mapfile.map
```

- **Internal Function**

MapType: `int`, MapSource: Internal Apache function

Here, the source is an internal Apache function. 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.

- **External Rewriting Program**

MapType: `prg`, MapSource: Unix filesystem path to valid regular file

Here the source is a program, not a map file. To create it you can use a language of your choice, but the result has to be an executable program (either object-code or a script with the magic cookie trick `'#!/path/to/interpreter'` as the first line).

This program is started once, when the Apache server is started, and then communicates with the rewriting engine via

its `stdin` and `stdout` file-handles. For each map-function lookup it will receive the key to lookup as a newline-terminated string on `stdin`. It then has to give back the looked-up value as a newline-terminated string on `stdout` or the four-character string ```NULL"` if it fails (*i.e.*, there is no corresponding value for the given key). A trivial program which will implement a 1:1 map (*i.e.*, `key == value`) could be:

External rewriting programs are not started if they're defined in a context that does not have `RewriteEngine` set to on.

```
#!/usr/bin/perl
$| = 1;
while (<STDIN>) {
    # ...put here any transformations or lookups...
    print $_;
}
```

But be very careful:

1. *``Keep it simple, stupid"* (KISS). If this program hangs, it will cause Apache to hang when trying to use the relevant rewrite rule.
2. A common mistake is to use buffered I/O on `stdout`. Avoid this, as it will cause a deadlock! ```$|=1"` is used above, to prevent this.
3. The `RewriteLock` directive can be used to define a lockfile which `mod_rewrite` can use to synchronize communication with the mapping program. By default no such synchronization takes place.

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 it is of course possible to **use** this map in per-directory context.

### Note

For plain text and DBM format files the looked-up keys are cached in-core until the `mtime` of the mapfile changes or the server does a restart. This way you can have map-functions in rules which are used for **every** request. This is no problem, because the external lookup only happens once!



<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
<b>Compatibility:</b>	MaxRedirects is no longer available in version 2.1 and later

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 are inherited.

Rules inherited from the parent scope are applied **after** rules specified in the child scope.

### **AllowAnyURI**

When `RewriteRule` is used in `VirtualHost` or server context with version 2.2.23 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.

### 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`. Not copying was the default until 2.2.22. In version 2.2.23 copying was the default. The flag to explicitly control it is available for Apache HTTP Server 2.2.24 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. On the first RewriteRule it is applied to the (%-decoded) [URL-path](#) of the request; subsequent patterns are applied to the output of the last matched RewriteRule.

### 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").

In **Directory** and htaccess context, the *Pattern* will initially be matched against the *filesystem* path, after removing the prefix that led the server to the current **RewriteRule** (e.g. "app1/index.html" or "index.html" depending on where the directives are defined).

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.
- When using the rewrite engine in `.htaccess` files the per-directory prefix (which always is the same for a specific directory) is automatically *removed* for the RewriteRule pattern matching and automatically *added* after any relative (not starting with a slash or protocol name) substitution encounters the end of a rule set. 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, this should never be necessary and is unsupported.

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 Nth 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 is **completely replaced** by the *Substitution* and the rewriting process continues until all rules have been applied, or it is explicitly terminated by a [L flag](#), or other flag which implies immediate termination, such as **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 <i>before</i> transformation. <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: <code>CO=NAME:VAL:domain[:lifetime[:path[:se</code> <a href="#">details ...</a>
discardpath DPI	Causes the PATH_INFO portion of the request to be discarded. <a href="#">details ...</a>

env E=[!]VAR[:VAL]	Causes an environment variable <i>VAR</i> to be value <i>VAL</i> if provided). The form <i>!VAR</i> causes environment variable <i>VAR</i> to be unset. <a href="#">details ...</a>
forbidden F	Returns a 403 FORBIDDEN response to the browser. <a href="#">details ...</a>
gone G	Returns a 410 GONE response to the client. <a href="#">details ...</a>
Handler H= <i>Content-handler</i>	Causes the resulting URI to be sent to the <i>Content-handler</i> for processing. <a href="#">details ...</a>
last L	Stop the rewriting process immediately and any more rules. Especially note caveats for and .htaccess context. <a href="#">details ...</a>
next N	Re-run the rewriting process, starting again at the first rule, using the result of the ruleset so far as a point. <a href="#">details ...</a>
nocase NC	Makes the pattern comparison case-insensitive.
noescape NE	Prevent mod_rewrite from applying hexcodes to special characters in the result of the rewrite.
nosubreq NS	Causes a rule to be skipped if the current request is an internal sub-request. <a href="#">details ...</a>
proxy P	Force the substitution URL to be internally proxied. <a href="#">details ...</a>
passthrough PT	Forces the resulting URI to be passed back to the mapping engine for processing of other URI translators, such as Alias or Redirect.
qsappend QSA	Appends any query string from the original request to any query string created in the rewrite target.
redirect R[= <i>code</i> ]	Forces an external redirect, optionally with an HTTP status code. <a href="#">details ...</a>
skip S= <i>num</i>	Tells the rewriting engine to skip the next <i>n</i> rules if the current rule matches. <a href="#">details ...</a>

type T= <i>MIME-type</i>	Force the <a href="#">MIME-type</a> of the target file to be type. <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 Substit
----- ^/somepath(.*) otherpath\$1	invalid, not supp
^/somepath(.*) otherpath\$1 [R]	invalid, not supp
^/somepath(.*) otherpath\$1 [P]	invalid, not supp
----- ^/somepath(.*) /otherpath\$1	/otherpath/pathir
^/somepath(.*) /otherpath\$1 [R]	http://thishost/c via external redi
^/somepath(.*) /otherpath\$1 [P]	doesn't make sens
----- ^/somepath(.*) http://thishost/otherpath\$1	/otherpath/pathir
^/somepath(.*) http://thishost/otherpath\$1 [R]	http://thishost/c via external redi
^/somepath(.*) http://thishost/otherpath\$1 [P]	doesn't make sens
-----	-----

```

^/somepath(.*) http://otherhost/otherpath$1 http://otherhost/
via external redi

^/somepath(.*) http://otherhost/otherpath$1 [R] http://otherhost/
via external redi
(the [R] flag is

^/somepath(.*) http://otherhost/otherpath$1 [P] http://otherhost/
via internal prox

```

**Inside per-directory configuration for /somepath  
(/physical/path/to/somepath/.htaccess, with  
RewriteBase /somepath)  
for request `GET /somepath/localpath/pathinfo`:**

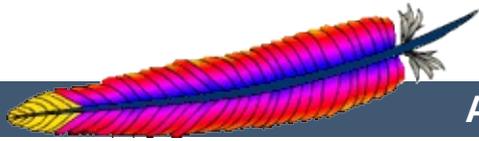
Given Rule	Resulting Substit
^localpath(.*) otherpath\$1	/somepath/otherpa
^localpath(.*) otherpath\$1 [R]	http://thishost/s via external redi
^localpath(.*) otherpath\$1 [P]	doesn't make sens
^localpath(.*) /otherpath\$1	/otherpath/pathir
^localpath(.*) /otherpath\$1 [R]	http://thishost/c via external redi
^localpath(.*) /otherpath\$1 [P]	doesn't make sens
^localpath(.*) http://thishost/otherpath\$1	/otherpath/pathir
^localpath(.*) http://thishost/otherpath\$1 [R]	http://thishost/c via external redi
^localpath(.*) http://thishost/otherpath\$1 [P]	doesn't make sens
^localpath(.*) http://otherhost/otherpath\$1	http://otherhost/ via external redi
^localpath(.*) http://otherhost/otherpath\$1 [R]	http://otherhost/ via external redi (the [R] flag is

```
^localpath(.*) http://otherhost/otherpath$1 [P] http://otherhost/  
via internal proxy
```

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_setenvif

- ┆
- ┆ Base
- ┆ setenvif\_module
- ┆ mod\_setenvif.c

## mod\_setenvif

MSIE mozilla

```
BrowserMatch ^Mozilla netscape  
BrowserMatch MSIE !netscape
```



## 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



```

:
:      SetEnvIfNoCase attribute regex [!]env-
:      variable[=value] [[!]env-
:      variable[=value]] ...
:      , , directory, .htaccess
Override : FileInfo
:      Base
:      mod_setenvif

```

```

SetEnvIfNoCase      SetEnvIf ,      .
:

```

```

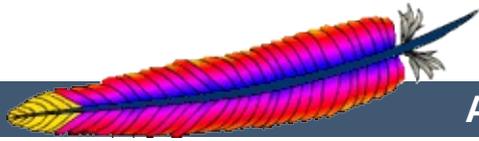
SetEnvIfNoCase Host Apache\.Org site=apache

```

```

HTTP      Host: Apache.Org, apache.org
site " apache".

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## 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
```



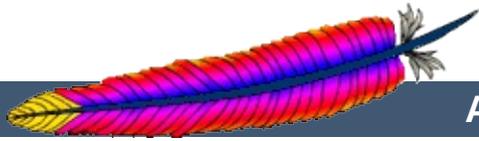
## LoadModule

```
LoadModule module filename
#
# Extension
# mod_so
```

LoadModule *filename* , *module*  
# *Module* *module* , *:*

```
LoadModule status_module modules/mod_status.so
```

ServerRoot modules .



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_speling

•

URL

•

Extension

•

speling\_module

•

mod\_speling.c

•  
)

,

- , "document not found ( )" .
- "" ,
- ,



CheckCaseOnly	
<code>:</code>	Limits the action of the speling module to case corrections
<code>:</code>	CheckCaseOnly on off
<code>:</code>	CheckCaseOnly Off
<code>:</code>	, , directory, .htaccess
<b>Override :</b>	Options
<code>:</code>	Extension
<code>:</code>	mod_speling

The documentation for this directive has not been translated yet. Please have a look at the English version.



## CheckSpelling

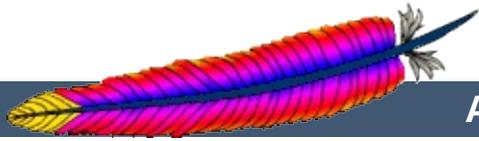
```

:
:      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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 v2/v3 and TLS v1 support for the Apache HTTP Server. It was contributed by Ralf S. Engeschall based on his mod\_ssl project and originally derived from work by Ben Laurie.

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.

<b>Variable Name:</b>	<b>Value Type:</b>	<b>Description:</b>
HTTPS	flag	HTTPS is being used.
SSL_PROTOCOL	string	The SSL protocol version (SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2)
SSL_SESSION_ID	string	The hex-encoded SSL session id
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 bits (actually used)
SSL_CIPHER_ALGKEYSIZE	number	Number of cipher bits (possible)
SSL_COMPRESS_METHOD	string	SSL compression method negotiated
SSL_VERSION_INTERFACE	string	The mod_ssl program version
SSL_VERSION_LIBRARY	string	The OpenSSL program

		version
SSL_CLIENT_M_VERSION	string	The version of the client certificate
SSL_CLIENT_M_SERIAL	string	The serial of the client certificate
SSL_CLIENT_S_DN	string	Subject DN in client's certificate
SSL_CLIENT_S_DN_x509	string	Component of client's Subject DN
SSL_CLIENT_I_DN	string	Issuer DN of client's certificate
SSL_CLIENT_I_DN_x509	string	Component of client's 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 the signature of client's certificate
SSL_CLIENT_A_KEY	string	Algorithm used for the public key of client's certificate
SSL_CLIENT_CERT	string	PEM-encoded client certificate
SSL_CLIENT_CERT_CHAIN_ <i>n</i>	string	PEM-encoded certificates in client certificate chain

SSL_CLIENT_VERIFY	string	NONE, SUCCESS, GENEROUS or FAILED: <i>reason</i>
SSL_SERVER_M_VERSION	string	The version of the server certificate
SSL_SERVER_M_SERIAL	string	The serial of the server certificate
SSL_SERVER_S_DN	string	Subject DN in server's certificate
SSL_SERVER_S_DN_x509	string	Component of server's Subject DN
SSL_SERVER_I_DN	string	Issuer DN of server's certificate
SSL_SERVER_I_DN_x509	string	Component of server's Issuer DN
SSL_SERVER_V_START	string	Validity of server's certificate (start time)
SSL_SERVER_V_END	string	Validity of server's certificate (end time)
SSL_SERVER_A_SIG	string	Algorithm used for the signature of server's certificate
SSL_SERVER_A_KEY	string	Algorithm used for the public key of server's certificate
SSL_SERVER_CERT	string	PEM-encoded server certificate
SSL_TLS_SNI	string	Contents of the SNI TLS extension (if supplied 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 an index to select a particular attribute. For example, where the server certificate subject DN included two OU fields, `SSL_SERVER_S_DN_OU_0` and `SSL_SERVER_S_DN_OU_1` could be used to reference each.

`SSL_CLIENT_V_REMAIN` is only available in version 2.1 and later.



## 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_CIPHER}x \"%r\" %b"
```

These formats even work without setting the `StdEnvVars` option of the [SSLOptions](#) directive.



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	<code>SSLCACertificateFile</code> <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-client.crt
```



<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/
```



**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-client.crl
```



<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>

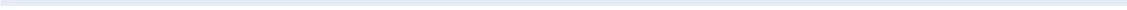
This directive sets the optional *all-in-one* file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate. Such 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 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>	mod_ssl
<b>Compatibility:</b>	ECC support is available in Apache 2.2.26 and later

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 up to three times (referencing different filenames) when an RSA, a DSA, and an ECC based server certificate is used in parallel.

Custom DH parameters and an EC curve name for ephemeral keys, can be added to end of the first file configured using [SSLCertificateFile](#). This is supported in version 2.2.30 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 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 the private key is encrypted, the pass phrase dialog is forced at startup time.

### **DH parameter interoperability with primes > 1024 bit**

Beginning with version 2.2.30, mod\_ssl makes use of standardized DH parameters with prime lengths of 2048, 3072,

4096, 6144 and 8192 bits (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.

### **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
<b>Compatibility:</b>	ECC support is available in Apache 2.2.26 and later

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 up to three times (referencing different filenames) when an RSA, a DSA, and an ECC based private key is used in parallel. 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.

### Example

```
SSLCertificateKeyFile  
/usr/local/apache2/conf/ssl.key/server.key
```



<b>Description:</b>	Cipher Suite available for negotiation in SSL handshake
<b>Syntax:</b>	SSLCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
<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 or Diffie-Hellman variants.
- *Authentication Algorithm:*  
RSA, Diffie-Hellman, DSS or none.
- *Cipher/Encryption Algorithm:*  
DES, Triple-DES, RC4, RC2, IDEA or none.
- *MAC Digest Algorithm:*  
MD5, SHA or SHA1.

An SSL cipher can also be an export cipher and is either a SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3). 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](#)).

<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)
<i>Authentication Algorithm:</i>	
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication
<i>Cipher Encoding Algorithm:</i>	
eNULL	No encoding
DES	DES encoding
3DES	Triple-DES encoding
RC4	RC4 encoding
RC2	RC2 encoding
IDEA	IDEA encoding
<i>MAC Digest Algorithm:</i>	
MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	SHA hash function
<i>Aliases:</i>	
SSLv2	all SSL version 2.0 ciphers
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
ADH	all ciphers using Anonymous Diffie-Hellman key exchange
DSS	all ciphers using DSS authentication
NULL	all ciphers using no encryption

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 (SSLv2, 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 2.2.30, null and export-grade ciphers are

always disabled, as `mod_ssl` unconditionally prepends any supplied cipher suite string with `!aNULL:!eNULL:!EXP:` 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 is

```
``ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP``
```

which means the following: first, remove from consideration any ciphers that do not authenticate, i.e. for SSL only the Anonymous Diffie-Hellman ciphers. Next, use ciphers using RC4 and RSA. Next include the high, medium and then the low security ciphers. Finally *pull* all SSLv2 and export ciphers to the end of the list.

```
$ openssl ciphers -v 'ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:
NULL-SHA          SSLv3 Kx=RSA      Au=RSA  Enc=None      M
NULL-MD5          SSLv3 Kx=RSA      Au=RSA  Enc=None      M
EDH-RSA-DES-CBC3-SHA  SSLv3 Kx=DH       Au=RSA  Enc=3DES(168) M
...
EXP-RC4-MD5       SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40)   M
EXP-RC2-CBC-MD5   SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40)   M
EXP-RC4-MD5       SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40)   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	

DES - CBC3 - MD5	SSLv2	RSA	RSA	3DES(168)	MD5	
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	
IDEA - CBC - MD5	SSLv2	RSA	RSA	IDEA(128)	MD5	
RC2 - CBC - MD5	SSLv2	RSA	RSA	RC2(128)	MD5	
RC4 - MD5	SSLv2	RSA	RSA	RC4(128)	MD5	
DES - CBC - SHA	SSLv3	RSA	RSA	DES(56)	SHA1	
RC4 - 64 - MD5	SSLv2	RSA	RSA	RC4(64)	MD5	
DES - CBC - MD5	SSLv2	RSA	RSA	DES(56)	MD5	
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
EXP - RC2 - CBC - MD5	SSLv2	RSA(512)	RSA	RC2(40)	MD5	export
EXP - RC4 - MD5	SSLv2	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-CBC3-SHA	SSLv3	DH	None	3DES(168)	SHA1	
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 2.2.24 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 versions 2.2.24 to 2.2.25.

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
<b>Compatibility:</b>	Available in Apache 2.1 and later, if using -engine flavor of OpenSSL 0.9.6, or OpenSSL 0.9.7 or later

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 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 <i>flag</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in Apache 2.1 and later, if using OpenSSL 0.9.7 or later

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 <i>flag</i>
<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>	Semaphore for internal mutual exclusion of operations
<b>Syntax:</b>	SSLMutex <i>type</i>
<b>Default:</b>	SSLMutex none
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures the SSL engine's semaphore (aka. lock) which is used for mutual exclusion of operations which have to be done in a synchronized way between the pre-forked Apache server processes. This directive can only be used in the global server context because it's only useful to have one global mutex. This directive is designed to closely match the [AcceptMutex](#) directive.

The following Mutex *types* are available:

- none | no  
This is the default where no Mutex is used at all. Use it at your own risk. But because currently the Mutex is mainly used for synchronizing write access to the SSL Session Cache you can live without it as long as you accept a sometimes garbled Session Cache. So it's not recommended to leave this the default. Instead configure a real Mutex.
- posixsem  
This is an elegant Mutex variant where a Posix Semaphore is used when possible. It is only available when the underlying platform and [APR](#) supports it.
- sysvsem  
This is a somewhat elegant Mutex variant where a SystemV IPC Semaphore is used when possible. It is possible to "leak"

SystemV semaphores if processes crash before the semaphore is removed. It is only available when the underlying platform and [APR](#) supports it.

- `sem`

This directive tells the SSL Module to pick the "best" semaphore implementation available to it, choosing between Posix and SystemV IPC, in that order. It is only available when the underlying platform and [APR](#) supports at least one of the 2.

- `pthread`

This directive tells the SSL Module to use Posix thread mutexes. It is only available if the underlying platform and [APR](#) supports it.

- `fcntl:/path/to/mutex`

This is a portable Mutex variant where a physical (lock-)file and the `fcntl()` function are used as the Mutex. Always use a local disk filesystem for `/path/to/mutex` and never a file residing on a NFS- or AFS-filesystem. It is only available when the underlying platform and [APR](#) supports it. Note: Internally, the Process ID (PID) of the Apache parent process is automatically appended to `/path/to/mutex` to make it unique, so you don't have to worry about conflicts yourself. Notice that this type of mutex is not available under the Win32 environment. There you *have* to use the semaphore mutex.

- `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. It is only available when the underlying platform and [APR](#) supports it.

- `file:/path/to/mutex`

This directive tells the SSL Module to pick the "best" file locking implementation available to it, choosing between `fcntl` and `flock`, in that order. It is only available when the underlying platform and [APR](#) supports at least one of the 2.

- `default` | `yes`

This directive tells the SSL Module to pick the default locking implementation as determined by the platform and [APR](#).

### Example

```
SSLMutex file:/usr/local/apache/logs/ssl_mutex
```



<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 -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/"`.

- 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 maybe not what the user expects, so enable this on a per-directory basis only, please.

### 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`, or `ECC`), which indicate for which server and algorithm it has to print the corresponding Pass Phrase to `stdout`. 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 protocol flavors
<b>Syntax:</b>	SSLProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProtocol 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. Clients then can only connect with one of the provided protocols.

The available (case-insensitive) *protocols* are:

- **SSLv2**  
This is the Secure Sockets Layer (SSL) protocol, version 2.0. It is the original SSL protocol as designed by Netscape Corporation. Though its use has been deprecated, because of weaknesses in the security of the protocol.
- **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. It's supported by almost all popular browsers.
- **TLSv1**  
This is the Transport Layer Security (TLS) protocol, version 1.0. It is the successor to SSLv3 and is defined in [RFC 2246](#).
- **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 ``+SSLv2 +SSLv3 +TLSv1" or - when using OpenSSL 1.0.1 and later - ``+SSLv2 +SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2", respectively.

### Example

```
# enable SSLv3 and all available TLSv1 flavors, but not SSLv2
SSLProtocol All -SSLv2
```



<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-bundle-remote-server.crt
```



<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>	File of concatenated PEM-encoded CA CRLs for Remote Server Auth
<b>Syntax:</b>	SSLProxyCAREvocationFile <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 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-  
bundle-remote-server.crl
```



<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 certificates CN field
<b>Syntax:</b>	SSLProxyCheckPeerCN on off
<b>Default:</b>	SSLProxyCheckPeerCN off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets whether the remote server certificates CN field is compared against the hostname of the request URL. If both are not equal a 502 status code (Bad Gateway) is sent.

### Example

```
SSLProxyCheckPeerCN on
```



## SSLProxyCheckPeerExpire Directive

<b>Description:</b>	Whether to check if remote server certificate is expired
<b>Syntax:</b>	SSLProxyCheckPeerExpire on off
<b>Default:</b>	SSLProxyCheckPeerExpire off
<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>	Cipher Suite available for negotiation in SSL proxy h
<b>Syntax:</b>	SSLProxyCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
<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 `<ProxyRequest>` 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
<b>Compatibility:</b>	Available in Apache 2.2.23 and later

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/ssl.crt/proxyCA.pem
```



<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/proxy.pem
```



<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.crt/
```



<b>Description:</b>	Configure usable SSL protocol flavors for proxy usage
<b>Syntax:</b>	SSLProxyProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProxyProtocol 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.



<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.

Note that even when certificate verification is enabled, [mod\\_ssl](#) does **not** check whether the commonName (hostname) attribute of the server certificate matches the hostname used to connect to the server. In other words, the proxy does not guarantee that the SSL connection to the backend server is "secure" beyond the fact that the certificate is signed by one of the CAs configured using the [SSLProxyCACertificatePath](#) and/or [SSLProxyCACertificateFile](#) directives. In order to get this check done please have a look at [SSLProxyCheckPeerCN](#) and [SSLProxyCheckPeerExpire](#) directives which are off by default.

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>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 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  $\emptyset$  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.

On some platforms like FreeBSD one can even control how the entropy is actually generated, i.e. by which system interrupts. More details one can find under `rndcontrol(8)` on those platforms. Alternatively, when your system lacks such a random device, you can use a tool like [EGD](#) (Entropy Gathering Daemon) and run its client program with the `exec:/path/to/program/` variant (see below) or use `egd:/path/to/egd-socket` (see below).

- `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
```



**Description:** Allow access only when an arbitrarily complex boolean expression is true

**Syntax:** `SSLRequire expression`

**Context:** directory, `.htaccess`

**Override:** AuthConfig

**Status:** Extension

**Module:** `mod_ssl`

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 implementation of `SSLRequire` is not thread safe. Using `SSLRequire` inside `.htaccess` files on a threaded [MPM](#) may cause random crashes.

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" "OID(" word ")"
        | word "=~" regex
        | word "!~" regex

wordlist ::= word
          | wordlist "," word

word     ::= digit
          | cstring
          | variable
          | function

digit    ::= [0-9]+
cstring  ::= "... "
variable ::= "%{" varname "}"
function ::= funcname "(" funcargs ")"

```

while for varname any variable from [Table 3](#) can be used. Finally for funcname the following functions are available:

- `file(filename)`  
This function takes one string argument and expands to the contents of the file. This is especially useful for matching this contents against a regular expression, etc.

Notice that *expression* is first parsed into an internal machine representation and then evaluated in a second step. Actually, in Global and Per-Server Class context *expression* is parsed at startup time and at runtime only the machine representation is executed. For Per-Directory context, specifically in a .htaccess context, this is different: here *expression* has to be parsed and immediately executed for every request.

### 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} >= 1 and %{TIME_WDAY} <= 5 \
and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20 ) \
or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/

```

The `OID()` function expects to find zero or more instances of the given OID in the client certificate, and compares the left-hand side string against the value of matching OID attributes. Every matching OID is checked, until a match is found.

*Standard CGI/1.0 and Apache variables:*

HTTP_USER_AGENT	PATH_INFO	AUTH
HTTP_REFERER	QUERY_STRING	SERV
HTTP_COOKIE	REMOTE_HOST	API_
HTTP_FORWARDED	REMOTE_IDENT	TIME
HTTP_HOST	IS_SUBREQ	TIME
HTTP_PROXY_CONNECTION	DOCUMENT_ROOT	TIME
HTTP_ACCEPT	SERVER_ADMIN	TIME
HTTP:headername	SERVER_NAME	TIME
THE_REQUEST	SERVER_PORT	TIME
REQUEST_METHOD	SERVER_PROTOCOL	TIME
REQUEST_SCHEME	REMOTE_ADDR	TIME
REQUEST_URI	REMOTE_USER	ENV:
REQUEST_FILENAME		

*SSL-related variables:*

HTTPS	SSL_CLIENT_M_VERSION	SSL
	SSL_CLIENT_M_SERIAL	SSL
SSL_PROTOCOL	SSL_CLIENT_V_START	SSL
SSL_SESSION_ID	SSL_CLIENT_V_END	SSL
SSL_CIPHER	SSL_CLIENT_S_DN	SSL
SSL_CIPHER_EXPORT	SSL_CLIENT_S_DN_C	SSL
SSL_CIPHER_ALGKEYSIZE	SSL_CLIENT_S_DN_ST	SSL
SSL_CIPHER_USEKEYSIZE	SSL_CLIENT_S_DN_L	SSL
SSL_VERSION_LIBRARY	SSL_CLIENT_S_DN_O	SSL

SSL_VERSION_INTERFACE	SSL_CLIENT_S_DN_OU	SSL.
	SSL_CLIENT_S_DN_CN	SSL.
	SSL_CLIENT_S_DN_T	SSL.
	SSL_CLIENT_S_DN_I	SSL.
	SSL_CLIENT_S_DN_G	SSL.
	SSL_CLIENT_S_DN_S	SSL.
	SSL_CLIENT_S_DN_D	SSL.
	SSL_CLIENT_S_DN_UID	SSL.
	SSL_CLIENT_S_DN_Email	SSL.
	SSL_CLIENT_I_DN	SSL.
	SSL_CLIENT_I_DN_C	SSL.
	SSL_CLIENT_I_DN_ST	SSL.
	SSL_CLIENT_I_DN_L	SSL.
	SSL_CLIENT_I_DN_O	SSL.
	SSL_CLIENT_I_DN_OU	SSL.
	SSL_CLIENT_I_DN_CN	SSL.
	SSL_CLIENT_I_DN_T	SSL.
	SSL_CLIENT_I_DN_I	SSL.
	SSL_CLIENT_I_DN_G	SSL.
	SSL_CLIENT_I_DN_S	SSL.
	SSL_CLIENT_I_DN_D	SSL.
	SSL_CLIENT_I_DN_UID	SSL.
	SSL_CLIENT_I_DN_Email	SSL.
	SSL_CLIENT_A_SIG	SSL.
	SSL_CLIENT_A_KEY	SSL.
	SSL_CLIENT_CERT	SSL.
	SSL_CLIENT_CERT_CHAIN_n	
	SSL_CLIENT_VERIFY	SSL.



<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 four 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.

- `shm:/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.
- `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.

## Examples

```
SSLSessionCache dbm:/usr/local/apache/logs/ssl_gcache_data
SSLSessionCache
shm:/usr/local/apache/logs/ssl_gcache_data(512000)
```



<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 2.2.28 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 2.2.30 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 2.2.30 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>	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 non default name based virtual host, non SNI clients are not allowed to access this particular virtual host. If set to on in the default name based virtual host, non SNI clients are not allowed to access any name based virtual host belonging to this IP / port combination.

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
<b>Compatibility:</b>	Available in Apache 2.0.51 and later

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
```



<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.

In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all browsers and level **optional\_no\_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

### 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

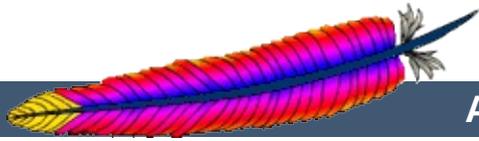
This directive sets how deeply mod\_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake 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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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
```



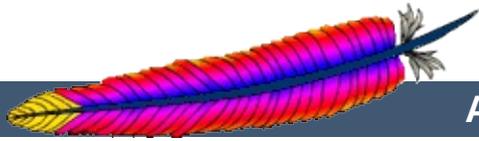
## ExtendedStatus

- ExtendedStatus On|Off
- ExtendedStatus Off
- Base
- mod\_status
- ExtendedStatus 1.3.2



- [:](#) 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.
- [:](#) SeeRequestTail On|Off
- [:](#) SeeRequestTail Off
- [:](#)
- [:](#) Base
- [:](#) mod\_status
- [:](#) Available in Apache 2.2.7 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 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>
```



<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 2.2.32 and later

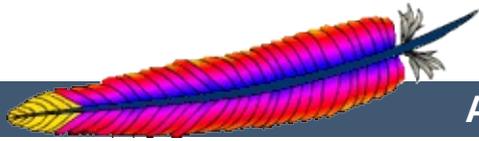
Whether to apply the inherited `Substitute` patterns first (on), or after the ones of the current context (off).

`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.

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# mod\_suexec

- [CGI](#)
- [Extension](#)
- [suexec\\_module](#)
- [mod\\_suexec.c](#)
- [2.0](#)

[suexec](#) CGI

[SuEXEC](#)

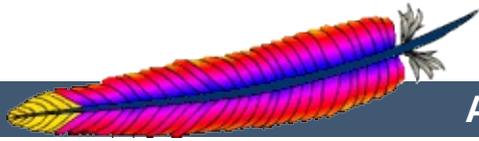


## SuexecUserGroup

- [: CGI](#)
- [: SuexecUserGroup \*User Group\*](#)
- [: ,](#)
- [: Extension](#)
- [: mod\\_suexec](#)
- [: SuexecUserGroup 2.0 .](#)

**SuexecUserGroup CGI** . CGI  
User . 1.3 VirtualHost Us  
Group .

```
SuexecUserGroup nobody nogroup
```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

# 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  
counter ) httpd 65536

( ip\_a  
. pid

httpd ( 10 )

65536 . (

( .)

(fork) pid  
32 .)

, pid . (pid  
pid . pid

32768 pid

65536 .  
, .)

).

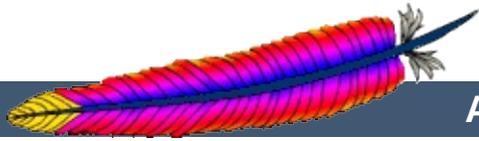
pid .

(

```

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 ,

```



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

## mod\_userdir

· ·

- 
- Base
- userdir\_module
- mod\_userdir.c

`http://example.com/~user/` .

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
```

. :

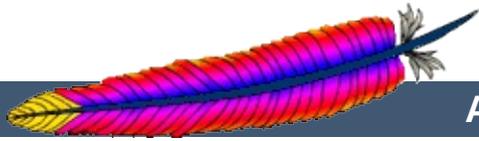
```
Userdir public_html /usr/web http://www.foo.com/
```

```
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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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

Previous releases of Apache have included a module which generates a 'clickstream' log of user activity on a site using cookies. This was called the "cookies" module, mod\_cookies. In Apache 1.2 and later this module has been renamed the "user tracking" module, mod\_usertrack. This module has been simplified and new directives added.



## Logging

Previously, the cookies module (now the user tracking module) did its own logging, using the `CookieLog` directive. In this release, this module does no logging at all. Instead, a configurable log format file should be used to log user click-streams. This is possible because the logging module now allows multiple log files. The cookie itself is logged by using the text `%{cookie}`n in the log file format. For example:

```
CustomLog logs/clickstream "%{cookie}n %r %t"
```

For backward compatibility the configurable log module implements the old `CookieLog` directive, but this should be upgraded to the above `CustomLog` directive.



(the following is from message  
<022701bda43d\$9d32bbb0\$1201a8c0@christian.office.sane.com>  
in the new-httpd archives)

From: "Christian Allen" <christian@sane.com>  
Subject: Re: Apache Y2K bug in mod\_usertrack.c  
Date: Tue, 30 Jun 1998 11:41:56 -0400

Did some work with cookies and dug up some info t

True, Netscape claims that the correct format NOW  
four digit dates do in fact work... for Netscape  
is. However, 3.x and below do NOT accept them.  
originally had a 2-digit standard, and then with  
probably a few complaints, changed to a four digit  
Fortunately, 4.x also understands the 2-digit form  
ensure that your expiration date is legible to th  
use 2-digit dates.

However, this does not limit expiration dates to  
an expiration year of "13", for example, it is in  
1913! In fact, you can use an expiration year of  
understood as "2037" by both MSIE and Netscape ve  
about versions previous to those). Not sure why  
particular year as its cut-off point, but my gues  
to UNIX's 2038 problem. Netscape/MSIE 4.x seem t  
2-digit years beyond that, at least until "50" fo  
understand up until about "70", but not for sure)

Summary: Mozilla 3.x and up understands two digi  
(2037). Mozilla 4.x understands up until at leas  
form, but also understands 4-digit years, which c  
9999. Your best bet for sending a long-life cook  
time late in the year "37".



<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 `foo.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, use `.foo.co.uk`).



<b>Description:</b>	Expiry time for the tracking cookie
<b>Syntax:</b>	CookieExpires <i>expiry-period</i>
<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.



<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 "-".



---

<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 only fully support CookieStyle Netscape.



## CookieTracking Directive

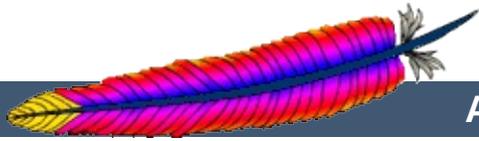
<b>Description:</b>	Enables tracking cookie
<b>Syntax:</b>	CookieTracking on off
<b>Default:</b>	CookieTracking off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

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.

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



## 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>
```

. .



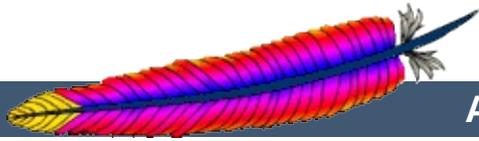
<i>operator</i>	
=    ==	<i>version / regex/</i>
~	<i>version regex</i>

```
<IfVersion = /^2.1.[01234]$/>
# ,
</IfVersion>
```

( ! ) .

```
<IfVersion !~ ^2.1.[01234]$>
#
</IfVersion>
```

*operator* = .



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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/bin/script.pl` in all cases:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/  
VirtualScriptAlias /never/found/%0/cgi-bin/
```

## 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/vhosts/%0
```

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/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

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/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/n/i/a/domain`

Alternatively you might use:

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/d/o/m/ain/di`

For IP-based virtual hosting you might use the following in your configuration file:

```
UseCanonicalName DNS
VirtualDocumentRootIP /usr/local/apache/vhosts/%1/%2/%3/%4/docs
VirtualScriptAliasIP /usr/local/apache/vhosts/%1/%2/%3/%4/cgi-
bin
```

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/vhosts/%2.0.%3.0
```

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 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`.



<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 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.



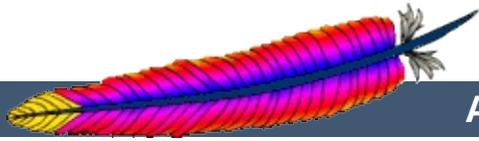
<b>Description:</b>	Dynamically configure the location of the cgi directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAliasIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAliasIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<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](#)



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 [DefaultType](#) 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).

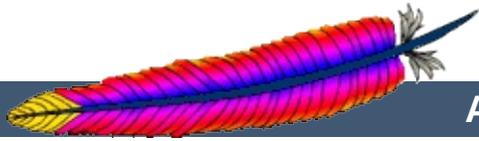
The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the `alias` module, which also indicates how a syntax error can be returned (note that the per-directory configuration 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Developer Documentation](#)

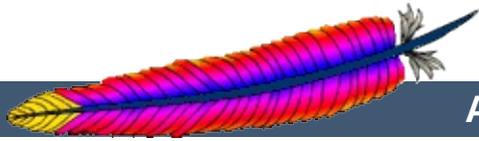
# Debugging Memory Allocation in APR

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Developer Documentation](#)

## Documenting Apache 2.0

Apache 2.0 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
 */
```

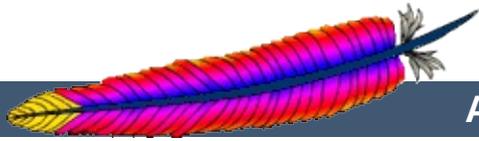
Doxygen uses a new HTML file for each package. The HTML files are named `{Name_of_library_header}.html`, so try to be concise with your names.

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](#)



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Developer Documentation](#)

# Apache 2.0 Hook Functions

## Warning

This document is still in development and may be partially out of date.

In general, a hook function is one that Apache 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.



## 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 *r, int n))
```

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, (request_rec *r, int n),
(r, n))
```

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, int n)
{
    ...
    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_something, (request_rec *r,
int n), (r, n), DECLINED)
```

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, (request_rec *r,
```

```
int n), (r, n), OK, DECLINED)
```

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, int n)
{
    ...
    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, NULL, NULL,
        APR_HOOK_MIDDLE);
}

module MODULE_VAR_EXPORT my_module =
{
    ...
    my_register_hooks /* register hooks */
};
```

### Controlling hook calling order

In the example above, we didn't use the three arguments in the hook registration function that control calling order. 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[] = { "mod_xyz.c",
        "mod_abc.c", NULL };

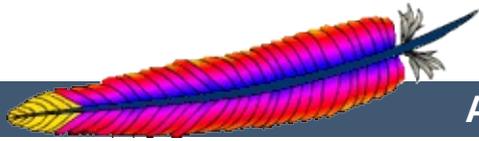
    ap_hook_do_something(my_something_doer, aszPre, NULL,
        APR_HOOK_MIDDLE);
}
```

Note that the sort used to achieve this is stable, so ordering set by APR\_HOOK\_ORDER is preserved, as far as is possible.

*Ben Laurie, 15th August 1999*

Licensed under the [Apache License, Version 2.0](#).

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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\_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\_open\_logs**

open any specified logs

**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**

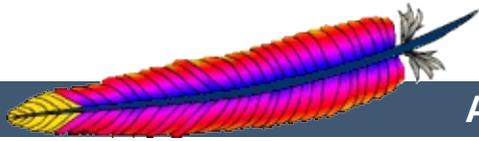
determine and/or set the doc type

---

Copyright 2017 The Apache Software Foundation.

Licensed under the [Apache License, Version 2.0](#).

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 `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:

```
switch (ap_satisfies(r)) {
case SATISFY_ALL:
case SATISFY_NOSPEC:
    if ((access_status = ap_run_access_checker(r)) != 0) {
        return decl_die(access_status, "check access", r);
    }

    if (ap_some_auth_required(r)) {
        if (((access_status = ap_run_check_user_id(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                ? "check user.  No user file?"
                : "perform authentication. AuthType not
                    r");
        }

        if (((access_status = ap_run_auth_checker(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                ? "check access.  No groups file?"
                : "perform authentication. AuthType not
                    r");
        }
    }
    break;

case SATISFY_ANY:
    if (((access_status = ap_run_access_checker(r)) != 0)) {
        if (!ap_some_auth_required(r)) {
            return decl_die(access_status, "check access", r);
        }
    }

    if (((access_status = ap_run_check_user_id(r)) != 0)
        || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r)
            ? "check user.  No user file?"
            : "perform authentication. AuthType not
                r");
    }

    if (((access_status = ap_run_auth_checker(r)) != 0)
        || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r)
            ? "check access.  No groups file?"
            : "perform authentication. AuthType not
```

```
        r);  
    }  
    }  
    break;  
}
```



## 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

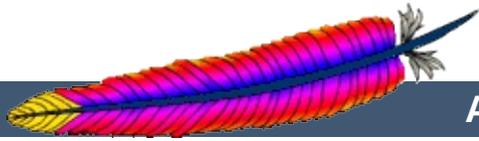
This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

### **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.



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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. So, the first list is `r->output_filters`, then `r->proto_output_filters`, and finally `r->connection->output_filters`. These correspond to the RESOURCE, PROTOCOL, and CONNECTION filters respectively. 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.

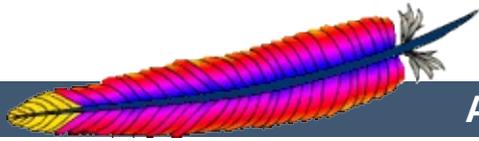
This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.



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.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



;



---

## (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) (HTTP) 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/2.2/glossary.html>

.

## Uniform Resource Identifier (URI)

. [RFC 2396](#) . URI

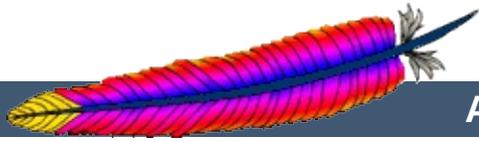
[URL](#) .

## (Virtual Hosting)

. IP IP . (name-based)  
IP .  
:

### X.509

(International Telecommunication Union, ITU-T) .  
SSL/TLS .  
: [SSL/TLS](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >

\_ .

A | B | C | D | E | F | G | H | I | K | L | M | N | O | P |  
R | S | T | U | V | W | X

- [AcceptFilter](#)
- [AcceptMutex](#)
- [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](#)
- [AllowOverride](#)
- [Anonymous](#)
- [Anonymous\\_LogEmail](#)
- [Anonymous\\_MustGiveEmail](#)
- [Anonymous\\_NoUserID](#)
- [Anonymous\\_VerifyEmail](#)
- [AuthBasicAuthoritative](#)
- [AuthBasicProvider](#)
- [AuthDBDUserPWQuery](#)
- [AuthDBDUserRealmQuery](#)
- [AuthDBMGroupFile](#)
- [AuthDBMType](#)
- [AuthDBMUserFile](#)
- [AuthDefaultAuthoritative](#)
- [AuthDigestAlgorithm](#)
- [AuthDigestDomain](#)
- [AuthDigestNcCheck](#)
- [AuthDigestNonceFormat](#)
- [AuthDigestNonceLifetime](#)
- [AuthDigestProvider](#)
- [AuthDigestQop](#)
- [AuthDigestShmemSize](#)
- [AuthGroupFile](#)
- [AuthLDAPBindAuthoritative](#)
- [AuthLDAPBindDN](#)
- [AuthLDAPBindPassword](#)
- [AuthLDAPCharsetConfig](#)
- [AuthLDAPCompareDNOnServer](#)
- [AuthLDAPDereferenceAliases](#)
- [AuthLDAPGroupAttribute](#)
- [AuthLDAPGroupAttributeIsDN](#)
- [AuthLDAPRemoteUserAttribute](#)

- [AuthLDAPRemoteUserIsDN](#)
- [AuthLDAPUrl](#)
- [AuthName](#)
- [<AuthnProviderAlias>](#)
- [AuthType](#)
- [AuthUserFile](#)
- [AuthzDBMAuthoritative](#)
- [AuthzDBMType](#)
- [AuthzDefaultAuthoritative](#)
- [AuthzGroupFileAuthoritative](#)
- [AuthzLDAPAuthoritative](#)
- [AuthzOwnerAuthoritative](#)
- [AuthzUserAuthoritative](#)
- [BalancerMember](#)
- [BrowserMatch](#)
- [BrowserMatchNoCase](#)
- [BufferedLogs](#)
- [CacheDefaultExpire](#)
- [CacheDirLength](#)
- [CacheDirLevels](#)
- [CacheDisable](#)
- [CacheEnable](#)
- [CacheFile](#)
- [CacheIgnoreCacheControl](#)
- [CacheIgnoreHeaders](#)
- [CacheIgnoreNoLastMod](#)
- [CacheIgnoreQueryString](#)
- [CacheIgnoreURLSessionIdentifiers](#)
- [CacheLastModifiedFactor](#)
- [CacheLock](#)
- [CacheLockMaxAge](#)
- [CacheLockPath](#)
- [CacheMaxExpire](#)
- [CacheMaxFileSize](#)

- [CacheMinFileSize](#)
- [CacheNegotiatedDocs](#)
- [CacheRoot](#)
- [CacheStoreNoStore](#)
- [CacheStorePrivate](#)
- [CGIDScriptTimeout](#)
- [CGIMapExtension](#)
- [CharsetDefault](#)
- [CharsetOptions](#)
- [CharsetSourceEnc](#)
- [CheckCaseOnly](#)
- [CheckSpelling](#)
- [ChrootDir](#)
- [ContentDigest](#)
- [CookieDomain](#)
- [CookieExpires](#)
- [CookieLog](#)
- [CookieName](#)
- [CookieStyle](#)
- [CookieTracking](#)
- [CoreDumpDirectory](#)
- [CustomLog](#)
- [Dav](#)
- [DavDepthInfinity](#)
- [DavGenericLockDB](#)
- [DavLockDB](#)
- [DavMinTimeout](#)
- [DBDExptime](#)
- [DBDKeep](#)
- [DBDMax](#)
- [DBDMin](#)
- [DBDParams](#)
- [DBDPersist](#)
- [DBDPrepareSQL](#)

- [DBDriver](#)
- [DefaultIcon](#)
- [DefaultLanguage](#)
- [DefaultType](#)
- [DeflateBufferSize](#)
- [DeflateCompressionLevel](#)
- [DeflateFilterNote](#)
- [DeflateInflateLimitRequestBody](#)
- [DeflateInflateRatioBurst](#)
- [DeflateInflateRatioLimit](#)
- [DeflateMemLevel](#)
- [DeflateWindowSize](#)
- [Deny](#)
- [<Directory>](#)
- [DirectoryIndex](#)
- [<DirectoryMatch>](#)
- [DirectorySlash](#)
- [DocumentRoot](#)
- [DumpIOInput](#)
- [DumpIOLogLevel](#)
- [DumpIOOutput](#)
- [EnableExceptionHook](#)
- [EnableMMAP](#)
- [EnableSendfile](#)
- [ErrorDocument](#)
- [ErrorLog](#)
- [Example](#)
- [ExpiresActive](#)
- [ExpiresByType](#)
- [ExpiresDefault](#)
- [ExtendedStatus](#)
- [ExtFilterDefine](#)
- [ExtFilterOptions](#)
- [FallbackResource](#)

- [FileETag](#)
- [<Files>](#)
- [<FilesMatch>](#)
- [FilterChain](#)
- [FilterDeclare](#)
- [FilterProtocol](#)
- [FilterProvider](#)
- [FilterTrace](#)
- [ForceLanguagePriority](#)
- [ForceType](#)
- [ForensicLog](#)
- [GprofDir](#)
- [GracefulShutdownTimeout](#)
- [Group](#)
- [Header](#)
- [HeaderName](#)
- [HostnameLookups](#)
- [HttpProtocolOptions](#)
- [IdentityCheck](#)
- [IdentityCheckTimeout](#)
- [<IfDefine>](#)
- [<IfModule>](#)
- [<IfVersion>](#)
- [ImapBase](#)
- [ImapDefault](#)
- [ImapMenu](#)
- [Include](#)
- [IndexHeadInsert](#)
- [IndexIgnore](#)
- [IndexOptions](#)
- [IndexOrderDefault](#)
- [IndexStyleSheet](#)
- [ISAPIAppendLogToErrors](#)
- [ISAPIAppendLogToQuery](#)

- [ISAPICacheFile](#)
- [ISAPIFakeAsync](#)
- [ISAPILogNotSupported](#)
- [ISAPIReadAheadBuffer](#)
- [KeepAlive](#)
- [KeepAliveTimeout](#)
- [LanguagePriority](#)
- [LDAPCacheEntries](#)
- [LDAPCacheTTL](#)
- [LDAPConnectionTimeout](#)
- [LDAPOpCacheEntries](#)
- [LDAPOpCacheTTL](#)
- [LDAPSharedCacheFile](#)
- [LDAPSharedCacheSize](#)
- [LDAPTrustedClientCert](#)
- [LDAPTrustedGlobalCert](#)
- [LDAPTrustedMode](#)
- [LDAPVerifyServerCert](#)
- [<Limit>](#)
- [<LimitExcept>](#)
- [LimitInternalRecursion](#)
- [LimitRequestBody](#)
- [LimitRequestFields](#)
- [LimitRequestFieldSize](#)
- [LimitRequestLine](#)
- [LimitXMLRequestBody](#)
- [Listen](#)
- [ListenBackLog](#)
- [LoadFile](#)
- [LoadModule](#)
- [<Location>](#)
- [<LocationMatch>](#)
- [LockFile](#)
- [LogFormat](#)

- [LogLevel](#)
- [MaxClients](#)
- [MaxKeepAliveRequests](#)
- [MaxMemFree](#)
- [MaxRanges](#)
- [MaxRequestsPerChild](#)
- [MaxRequestsPerThread](#)
- [MaxSpareServers](#)
- [MaxSpareThreads](#)
- [MaxThreads](#)
- [MCacheMaxObjectCount](#)
- [MCacheMaxObjectSize](#)
- [MCacheMaxStreamingBuffer](#)
- [MCacheMinObjectSize](#)
- [MCacheRemovalAlgorithm](#)
- [MCacheSize](#)
- [MergeTrailers](#)
- [MetaDir](#)
- [MetaFiles](#)
- [MetaSuffix](#)
- [MimeMagicFile](#)
- [MinSpareServers](#)
- [MinSpareThreads](#)
- [MMapFile](#)
- [ModMimeUsePathInfo](#)
- [MultiviewsMatch](#)
- [NameVirtualHost](#)
- [NoProxy](#)
- [NWSSLTrustedCerts](#)
- [NWSSLUpgradeable](#)
- [Options](#)
- [Order](#)
- [PassEnv](#)
- [PidFile](#)

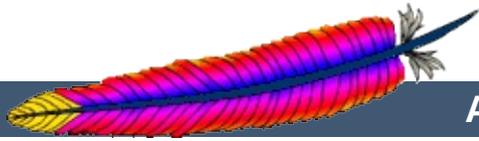
- [Protocol](#)
- [ProtocolEcho](#)
- [<Proxy>](#)
- [ProxyBadHeader](#)
- [ProxyBlock](#)
- [ProxyDomain](#)
- [ProxyErrorOverride](#)
- [ProxyFtpDirCharset](#)
- [ProxyIOBufferSize](#)
- [<ProxyMatch>](#)
- [ProxyMaxForwards](#)
- [ProxyPass](#)
- [ProxyPassInterpolateEnv](#)
- [ProxyPassMatch](#)
- [ProxyPassReverse](#)
- [ProxyPassReverseCookieDomain](#)
- [ProxyPassReverseCookiePath](#)
- [ProxyPreserveHost](#)
- [ProxyReceiveBufferSize](#)
- [ProxyRemote](#)
- [ProxyRemoteMatch](#)
- [ProxyRequests](#)
- [ProxySCGIInternalRedirect](#)
- [ProxySCGISendfile](#)
- [ProxySet](#)
- [ProxyStatus](#)
- [ProxyTimeout](#)
- [ProxyVia](#)
- [ReadmeName](#)
- [ReceiveBufferSize](#)
- [Redirect](#)
- [RedirectMatch](#)
- [RedirectPermanent](#)
- [RedirectTemp](#)

- [RegisterHttpMethod](#)
- [RemoveCharset](#)
- [RemoveEncoding](#)
- [RemoveHandler](#)
- [RemoveInputFilter](#)
- [RemoveLanguage](#)
- [RemoveOutputFilter](#)
- [RemoveType](#)
- [RequestHeader](#)
- [RequestReadTimeout](#)
- [Require](#)
- [RewriteBase](#)
- [RewriteCond](#)
- [RewriteEngine](#)
- [RewriteLock](#)
- [RewriteLog](#)
- [RewriteLogLevel](#)
- [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](#)
- [SetEnv](#)
- [SetEnvIf](#)
- [SetEnvIfNoCase](#)
- [SetHandler](#)
- [SetInputFilter](#)
- [SetOutputFilter](#)
- [SSIEnableAccess](#)
- [SSIEndTag](#)
- [SSIErrorMsg](#)
- [SSIETag](#)
- [SSILastModified](#)
- [SSIStartTag](#)
- [SSITimeFormat](#)
- [SSIUndefinedEcho](#)
- [SSLCACertificateFile](#)
- [SSLCACertificatePath](#)
- [SSLCADNRequestFile](#)
- [SSLCADNRequestPath](#)
- [SSLCARevocationFile](#)
- [SSLCARevocationPath](#)
- [SSLCertificateChainFile](#)
- [SSLCertificateFile](#)
- [SSLCertificateKeyFile](#)
- [SSLCipherSuite](#)

- [SSLCompression](#)
- [SSLCryptoDevice](#)
- [SSLEngine](#)
- [SSLFIPS](#)
- [SSLHonorCipherOrder](#)
- [SSLInsecureRenegotiation](#)
- [SSLMutex](#)
- [SSLOptions](#)
- [SSLPassPhraseDialog](#)
- [SSLProtocol](#)
- [SSLProxyCACertificateFile](#)
- [SSLProxyCACertificatePath](#)
- [SSLProxyCARevocationFile](#)
- [SSLProxyCARevocationPath](#)
- [SSLProxyCheckPeerCN](#)
- [SSLProxyCheckPeerExpire](#)
- [SSLProxyCipherSuite](#)
- [SSLProxyEngine](#)
- [SSLProxyMachineCertificateChainFile](#)
- [SSLProxyMachineCertificateFile](#)
- [SSLProxyMachineCertificatePath](#)
- [SSLProxyProtocol](#)
- [SSLProxyVerify](#)
- [SSLProxyVerifyDepth](#)
- [SSLRandomSeed](#)
- [SSLRenegBufferSize](#)
- [SSLRequire](#)
- [SSLRequireSSL](#)
- [SSLSessionCache](#)
- [SSLSessionCacheTimeout](#)
- [SSLSessionTicketKeyFile](#)
- [SSLSessionTickets](#)
- [SSLStrictSNIVHostCheck](#)
- [SSLUserName](#)

- [SSLVerifyClient](#)
- [SSLVerifyDepth](#)
- [StartServers](#)
- [StartThreads](#)
- [Substitute](#)
- [SubstituteInheritBefore](#)
- [Suexec](#)
- [SuexecUserGroup](#)
- [ThreadLimit](#)
- [ThreadsPerChild](#)
- [ThreadStackSize](#)
- [TimeOut](#)
- [TraceEnable](#)
- [TransferLog](#)
- [TypesConfig](#)
- [UnsetEnv](#)
- [UseCanonicalName](#)
- [UseCanonicalPhysicalPort](#)
- [User](#)
- [UserDir](#)
- [VirtualDocumentRoot](#)
- [VirtualDocumentRootIP](#)
- [<VirtualHost>](#)
- [VirtualScriptAlias](#)
- [VirtualScriptAliasIP](#)
- [Win32DisableAcceptEx](#)
- [XBitHack](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) >



...

A	B	C	D	E	F	G	H
I	K	L	M	N	O	P	R
S	T	U	V	W	X		

<b>s</b>		<b>C</b> Core
<b>v</b>		<b>M</b> MPM
<b>d</b> directory		<b>B</b> Base
<b>h</b> .htaccess		<b>E</b> Extension
		<b>X</b> Experimental

<a href="#">AcceptFilter <i>protocol accept filter</i></a>	
Configures optimizations for a Protocol's Listener Sockets	
<a href="#">AcceptMutex <i>Default method</i></a>	Default
Method that Apache uses to serialize multiple children accepting requests on network socket	
<a href="#">AcceptPathInfo <i>On Off Default</i></a>	Default
Resources accept trailing pathname information	
<a href="#">AccessFileName <i>filename [filename] ...</i></a>	.htaccess
Name of the distributed configuration file	
<a href="#">Action <i>action-type cgi-script [virtual]</i></a>	
content-type CGI	
<a href="#">AddAlt <i>string file [file] ...</i></a>	
<a href="#">AddAltByEncoding <i>string MIME-encoding [MIME-encoding] ...</i></a>	
MIME-encoding	
<a href="#">AddAltByType <i>string MIME-type [MIME-type]</i></a>	
...	

<a href="#">MIME content-type</a>	
<a href="#">AddCharset <i>charset_extension</i> [<i>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</i> [<i>file</i>] ...</a>	
<a href="#">AddEncoding <i>MIME-enc_extension</i> [<i>extension</i>] ...</a>	
Maps the given filename extensions to the specified encoding type	
<a href="#">AddHandler <i>handler-name_extension</i> [<i>extension</i>] ...</a>	
Maps the filename extensions to the specified handler	
<a href="#">AddIcon <i>icon_name</i> [<i>name</i>] ...</a>	
<a href="#">AddIconByEncoding <i>icon MIME-encoding</i> [<i>MIME-encoding</i>] ...</a>	
MIME content-encoding	
<a href="#">AddIconByType <i>icon MIME-type</i> [<i>MIME-type</i>] ...</a>	
MIME content-type	
<a href="#">AddInputFilter <i>filter</i>[:<i>filter...</i>] <i>extension</i> [<i>extension</i>] ...</a>	
Maps filename extensions to the filters that will process client requests	
<a href="#">AddLanguage <i>MIME-lang_extension</i> [<i>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</i>[:<i>filter...</i>] <i>extension</i> [<i>extension</i>] ...</a>	
Maps filename extensions to the filters that will process responses from the server	
<a href="#">AddOutputFilterByType <i>filter</i>[:<i>filter...</i>] <i>MIME-type</i> [<i>MIME-type</i>] ...</a>	
assigns an output filter to a particular MIME-type	

<a href="#">AddType <i>MIME-type extension [extension] ...</i></a>	
Maps the given filename extensions onto the specified content type	
<a href="#">Alias <i>URL-path file-path directory-path</i></a>	
URL	
<a href="#">AliasMatch <i>regex file-path directory-path</i></a>	
URL	
<a href="#">Allow from all <i>host env=env-variable</i></a> <a href="#">[<i>host env=env-variable</i>] ...</a>	
<a href="#">AllowCONNECT <i>port [port] ...</i></a>	443 563
Ports that are allowed to CONNECT through the proxy	
<a href="#">AllowEncodedSlashes On Off NoDecode</a>	Off
Determines whether encoded path separators in URLs are allowed to be passed through	
<a href="#">AllowOverride All None <i>directive-type</i></a> <a href="#">[<i>directive-type</i>] ...</a>	All
Types of directives that are allowed in .htaccess files	
<a href="#">Anonymous <i>user [user] ...</i></a>	
<a href="#">Anonymous_ LogEmail On Off</a>	On
<a href="#">Anonymous_ MustGiveEmail On Off</a>	On
<a href="#">Anonymous_ NoUserID On Off</a>	Off
<a href="#">Anonymous_ VerifyEmail On Off</a>	Off
<a href="#">AuthBasicAuthoritative On Off</a>	On
<a href="#">AuthBasicProvider On Off <i>provider-name</i></a> <a href="#">[<i>provider-name</i>] ...</a>	On
<a href="#">AuthDBDUserPWQuery <i>query</i></a>	
SQL query to look up a password for a user	
<a href="#">AuthDBDUserRealmQuery <i>query</i></a>	
SQL query to look up a password hash for a user and realm.	

<a href="#">AuthDBMGroupFile <i>file-path</i></a>	
<a href="#">AuthDBMType</a> <a href="#">default SDBM GDBM NDBM DB</a>	default
<a href="#">AuthDBMUserFile <i>file-path</i></a>	
<a href="#">AuthDefaultAuthoritative On Off</a>	On
<a href="#">AuthDigestAlgorithm MD5 MD5-sess</a> digest authentication challenge response hash	MD5
<a href="#">AuthDigestDomain <i>URI [URI] ...</i></a> digest authentication URI	
<a href="#">AuthDigestNcCheck On Off</a> nonce-count	Off
<a href="#">AuthDigestNonceFormat <i>format</i></a> nonce	
<a href="#">AuthDigestNonceLifetime <i>seconds</i></a> nonce	300
<a href="#">AuthDigestProvider On Off <i>provider-name</i></a> <a href="#">[<i>provider-name</i>] ...</a>	On
<a href="#">AuthDigestQop none auth auth-int [<i>auth auth-int</i>]</a> digest authentication (quality-of-protection) .	auth
<a href="#">AuthDigestShmemSize <i>size</i></a>	1000
<a href="#">AuthGroupFile <i>file-path</i></a>	
<a href="#">AuthLDAPBindAuthoritative<i>off on</i></a> Determines if other authentication providers are used when a user can be mapped to a DN cannot successfully bind with the user's credentials.	on
<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>	

<p>Password used in conjunction with the bind DN</p>	
<p><a href="#">AuthLDAPCharsetConfig <i>file-path</i></a></p> <p>Language to charset conversion configuration file</p>	
<p><a href="#">AuthLDAPCompareDNOnServer on off</a></p> <p>Use the LDAP server to compare the DN's</p>	on
<p><a href="#">AuthLDAPDereferenceAliases never searching finding always</a></p> <p>When will the module de-reference aliases</p>	Always
<p><a href="#">AuthLDAPGroupAttribute <i>attribute</i></a></p> <p>LDAP attributes used to check for group membership</p>	member uniquememb +
<p><a href="#">AuthLDAPGroupAttributeIsDN on off</a></p> <p>Use the DN of the client username when checking for group membership</p>	on
<p><a href="#">AuthLDAPRemoteUserAttribute <i>uid</i></a></p> <p>Use the value of the attribute returned during the user query to set the REMOTE_USER environment variable</p>	
<p><a href="#">AuthLDAPRemoteUserIsDN on off</a></p> <p>Use the DN of the client username to set the REMOTE_USER environment variable</p>	off
<p><a href="#">AuthLDAPUrl <i>url</i> [NONE SSL TLS STARTTLS]</a></p> <p>URL specifying the LDAP search parameters</p>	
<p><a href="#">AuthName <i>auth-domain</i></a></p> <p>Authorization realm for use in HTTP authentication</p>	
<p><a href="#">&lt;AuthnProviderAlias <i>baseProvider Alias</i>&gt; ...</a></p> <p><a href="#">&lt;/AuthnProviderAlias&gt;</a></p> <p>Enclose a group of directives that represent an extension of a base authentication provider by the specified alias</p>	
<p><a href="#">AuthType Basic Digest</a></p> <p>Type of user authentication</p>	
<p><a href="#">AuthUserFile <i>file-path</i></a></p>	
<p><a href="#">AuthzDBMAuthoritative On Off</a></p>	On
<p><a href="#">AuthzDBMType</a></p> <p><a href="#">default SDBM GDBM NDBM DB</a></p>	default

<a href="#">AuthzDefaultAuthoritative On Off</a>	On
<a href="#">AuthzGroupFileAuthoritative On Off</a>	On
<a href="#">AuthzLDAPAuthoritative on off</a> Prevent other authentication modules from authenticating the user if this one fails	on
<a href="#">AuthzOwnerAuthoritative On Off</a>	On
<a href="#">AuthzUserAuthoritative On Off</a>	On
<a href="#">BalancerMember [<i>balancerurl</i>] url [<i>key=value</i> [<i>key=value ...</i>]]</a> Add a member to a load balancing group	
<a href="#">BrowserMatch <i>regex</i> [!]env-variable[=<i>value</i>] [!]env-variable[=<i>value</i>] ...</a> HTTP User-Agent	
<a href="#">BrowserMatchNoCase <i>regex</i> [!]env-variable[=<i>value</i>] [!]env-variable[=<i>value</i>] ...</a> User-Agent	
<a href="#">BufferedLogs On Off</a> Buffer log entries in memory before writing to disk	Off
<a href="#">CacheDefaultExpire <i>seconds</i></a>	3600 (one hour)
<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> [<i>file-path</i>] ...</a>	
<a href="#">CacheIgnoreCacheControl On Off</a>	Off

<a href="#">CacheIgnoreHeaders <i>header-string [header-string] ...</i></a> HTTP ()	None
<a href="#">CacheIgnoreNoLastMod On Off</a> Last Modified .	Off
<a href="#">CacheIgnoreQueryString On Off</a> Ignore query string when caching	Off
<a href="#">CacheIgnoreURLSessionIdentifiers <i>identifier [identifier] ...</i></a> Ignore defined session identifiers encoded in the URL when caching	None
<a href="#">CacheLastModifiedFactor <i>float</i></a> LastModified .	0.1
<a href="#">CacheLock <i>on off</i></a> Enable the thundering herd lock.	off
<a href="#">CacheLockMaxAge <i>integer</i></a> Set the maximum possible age of a cache lock.	5
<a href="#">CacheLockPath <i>directory</i></a> Set the lock path directory.	/tmp/mod_cache-lock
<a href="#">CacheMaxExpire <i>seconds</i></a>	86400 ()
<a href="#">CacheMaxFileSize <i>bytes</i></a> ( )	1000000
<a href="#">CacheMinFileSize <i>bytes</i></a> ( )	1
<a href="#">CacheNegotiatedDocs On Off</a> Allows content-negotiated documents to be cached by proxy servers	Off
<a href="#">CacheRoot <i>directory</i></a> root	
<a href="#">CacheStoreNoStore On Off</a> Attempt to cache requests or responses that have been marked as no-store.	Off
<a href="#">CacheStorePrivate On Off</a> Attempt to cache responses that the server has marked as private	Off
<a href="#">CGIDScriptTimeout <i>time[s ms]</i></a> The length of time to wait for more output from the CGI program	
<a href="#">CGIMapExtension <i>cgi-path .extension</i></a>	

Technique for locating the interpreter for CGI scripts <a href="#">CharsetDefault <i>charset</i></a>	
<a href="#">CharsetOptions <i>option [option] ...</i></a>	DebugLevel=0 NoImp
<a href="#">CharsetSourceEnc <i>charset</i></a>	
<a href="#">CheckCaseOnly on off</a> Limits the action of the spelling module to case corrections	Off
<a href="#">CheckSpelling on off</a>	Off
<a href="#">ChrootDir <i>/path/to/directory</i></a> Directory for apache to run chroot(8) after startup.	
<a href="#">ContentDigest On Off</a> Enables the generation of Content-MD5 HTTP Response headers	Off
<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="#">CookieLog <i>filename</i></a>	
<a href="#">CookieName <i>token</i></a> Name of the tracking cookie	Apache
<a href="#">CookieStyle</a> <a href="#">Netscape Cookie Cookie2 RFC2109 RFC2965</a> Format of the cookie header field	Netscape
<a href="#">CookieTracking on off</a> Enables tracking cookie	off
<a href="#">CoreDumpDirectory <i>directory</i></a> Directory where Apache attempts to switch before dumping core	
<a href="#">CustomLog <i>file pipe format nickname [env=[!]<i>environment-variable</i>]</i></a>	
<a href="#">Dav On Off <i>provider-name</i></a> WebDAV HTTP	Off

<a href="#">DavDepthInfinity</a> <i>on off</i>	off
PROPFIND Depth: Infinity	
<a href="#">DavGenericLockDB</a> <i>file-path</i>	
Location of the DAV lock database	
<a href="#">DavLockDB</a> <i>file-path</i>	
DAV	
<a href="#">DavMinTimeout</a> <i>seconds</i>	0
DAV	
<a href="#">DBDExptime</a> <i>time-in-seconds</i>	300
Keepalive time for idle connections	
<a href="#">DBDKeep</a> <i>number</i>	2
Maximum sustained number of connections	
<a href="#">DBDMax</a> <i>number</i>	10
Maximum number of connections	
<a href="#">DBDMin</a> <i>number</i>	1
Minimum number of connections	
<a href="#">DBDParams</a> <i>param1=value1[,param2=value2]</i>	
Parameters for database connection	
<a href="#">DBDPersist</a> <i>On Off</i>	
Whether to use persistent connections	
<a href="#">DBDPrepareSQL</a> <i>"SQL statement" label</i>	
Define an SQL prepared statement	
<a href="#">DBDriver</a> <i>name</i>	
Specify an SQL driver	
<a href="#">DefaultIcon</a> <i>url-path</i>	
<a href="#">DefaultLanguage</a> <i>MIME-lang</i>	
Sets all files in the given scope to the specified language	
<a href="#">DefaultType</a> <i>MIME-type none</i>	text/plain
MIME content-type that will be sent if the server cannot determine a type in any other way	
<a href="#">DeflateBufferSize</a> <i>value</i>	8096
zlib	
<a href="#">DeflateCompressionLevel</a> <i>value</i>	
<a href="#">DeflateFilterNote</a> <i>[type] notename</i>	

<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> <i>env=env-variable</i></a> <a href="#">[<i>host</i> <i>env=env-variable</i>] ...</a>	
<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-directories and their contents	
<a href="#">DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...</a>	index.html
<a href="#">&lt;DirectoryMatch <i>regex</i>&gt; ... &lt;/DirectoryMatch&gt;</a>	
Enclose directives that apply to file-system directories matching a regular expression and their subdirectories	
<a href="#">DirectorySlash On Off</a>	On
<a href="#">DocumentRoot <i>directory-path</i></a>	/usr/local/apache/h +
Directory that forms the main document tree visible from the web	
<a href="#">DumpIOInput On Off</a>	Off
Dump all input data to the error log	
<a href="#">DumpIOLogLevel <i>level</i></a>	debug
Controls the logging level of the DumpIO output	
<a href="#">DumpIOOutput On Off</a>	Off
Dump all output data to the error log	
<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>	On
Use the kernel sendfile support to deliver files to the client	
<a href="#">ErrorDocument <i>error-code document</i></a>	
What the server will return to the client in case of an error	
<a href="#">ErrorLog <i>file-path syslog[:facility]</i></a>	logs/error_log (Uni +
Location where the server will log errors	
<a href="#">Example</a>	
API	
<a href="#">ExpiresActive On Off</a>	
Expires	
<a href="#">ExpiresByType <i>MIME-type &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
<a href="#">ExtFilterDefine <i>filtername parameters</i></a>	
<a href="#">ExtFilterOptions <i>option [option] ...</i></a>	DebugLevel=0 NoLog +
<code>mod_ext_filter</code>	
<a href="#">FallbackResource disabled   <i>local-url</i></a>	
Define a default URL for requests that don't map to a file	
<a href="#">FileETag <i>component ...</i></a>	INode 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 [+!@] <i>filter-name ...</i></a>	
Configure the filter chain	
<a href="#">FilterDeclare <i>filter-name [type]</i></a>	
Declare a smart filter	
<a href="#">FilterProtocol <i>filter-name [provider-name]</i></a>	

<a href="#"><u>proto-flags</u></a>	
Deal with correct HTTP protocol handling	
<a href="#"><u>FilterProvider <i>filter-name</i> <i>provider-name</i> [req resp env]=<i>dispatch</i> <i>match</i></u></a>	
Register a content filter	
<a href="#"><u>FilterTrace <i>filter-name</i> <i>level</i></u></a>	
Get debug/diagnostic information from <a href="#"><u>mod_filter</u></a>	
<a href="#"><u>ForceLanguagePriority None Prefer Fallback [Prefer Fallback]</u></a>	Prefer
Action to take if a single acceptable document is not found	
<a href="#"><u>ForceType <i>MIME-type</i> None</u></a>	
Forces all matching files to be served with the specified MIME content-type	
<a href="#"><u>ForensicLog <i>filename</i> <i>pipe</i></u></a>	
Sets filename of the forensic log	
<a href="#"><u>GprofDir <i>/tmp/gprof/</i><i>/tmp/gprof/%</i></u></a>	
Directory to write gmon.out profiling data to.	
<a href="#"><u>GracefulShutdownTimeout <i>seconds</i></u></a>	
Specify a timeout after which a gracefully shutdown server will exit.	
<a href="#"><u>Group <i>unix-group</i></u></a>	#-1
Group under which the server will answer requests	
<a href="#"><u>Header [<i>condition</i>] set append add unset echo <i>header</i> [<i>value</i>] [<i>early</i> env=[!]<i>variable</i>]</u></a>	
HTTP	
<a href="#"><u>HeaderName <i>filename</i></u></a>	
<a href="#"><u>HostnameLookups On Off Double</u></a>	Off
Enables DNS lookups on client IP addresses	
<a href="#"><u>HttpProtocolOptions [Strict Unsafe] [RegisteredMethods LenientMethods] [Allow0.9 Require1.0]</u></a>	Strict LenientMetho +
Modify restrictions on HTTP Request Messages	
<a href="#"><u>IdentityCheck On Off</u></a>	Off
RFC 1413	
<a href="#"><u>IdentityCheckTimeout <i>seconds</i></u></a>	30
ident	

<a href="#">&lt;IfDefine [!]parameter-name&gt; ... &lt;/IfDefine&gt;</a>	
Encloses directives that will be processed only if a test is true at startup	
<a href="#">&lt;IfModule [!]module-file module-identifier&gt; ...</a> <a href="#">&lt;/IfModule&gt;</a>	
Encloses directives that are processed conditional on the presence or absence of a specific	
<a href="#">&lt;IfVersion [!]operator version&gt; ...</a> <a href="#">&lt;/IfVersion&gt;</a>	
<a href="#">ImapBase map referer URL</a> base	http://servername/
<a href="#">ImapDefault error nocontent map referer URL</a>	nocontent
<a href="#">ImapMenu</a> <a href="#">none formatted semiformatted unformatted</a>	
<a href="#">Include file-path directory-path</a>	
Includes other configuration files from within the server configuration files	
<a href="#">IndexHeadInsert "markup ..."</a>	
Inserts text in the HEAD section of an index page.	
<a href="#">IndexIgnore file [file] ...</a>	
<a href="#">IndexOptions [+ -]option [[+ -]option] ...</a>	
<a href="#">IndexOrderDefault Ascending Descending Name Date Size Description</a>	Ascending Name
<a href="#">IndexStyleSheet url-path</a> CSS	
<a href="#">ISAPIAppendLogToErrors on off</a> ISAPI exntension HSE_APPEND_LOG_PARAMETER	off
<a href="#">ISAPIAppendLogToQuery on off</a> ISAPI exntension HSE_APPEND_LOG_PARAMETER	on
<a href="#">ISAPICacheFile file-path [file-path] ...</a> ISAPI .dll	

<a href="#">ISAPIFakeAsync on off</a>	off
ISAPI	
<a href="#">ISAPILogNotSupported on off</a>	off
ISAPI extension	
<a href="#">ISAPIReadAheadBuffer size</a>	49152
ISAPI extension (read ahead buffer)	
<a href="#">KeepAlive On Off</a>	On
Enables HTTP persistent connections	
<a href="#">KeepAliveTimeout seconds</a>	5
Amount of time the server will wait for subsequent requests on a persistent connection	
<a href="#">LanguagePriority MIME-lang [MIME-lang] ...</a>	
The precedence of language variants for cases where the client does not express a prefer	
<a href="#">LDAPCacheEntries number</a>	1024
Maximum number of entries in the primary LDAP cache	
<a href="#">LDAPCacheTTL seconds</a>	600
Time that cached items remain valid	
<a href="#">LDAPConnectionTimeout seconds</a>	
Specifies the socket connection timeout in seconds	
<a href="#">LDAPOpCacheEntries number</a>	1024
Number of entries used to cache LDAP compare operations	
<a href="#">LDAPOpCacheTTL seconds</a>	600
Time that entries in the operation cache remain valid	
<a href="#">LDAPSharedCacheFile directory-path/filename</a>	
Sets the shared memory cache file	
<a href="#">LDAPSharedCacheSize bytes</a>	500000
Size in bytes of the shared-memory cache	
<a href="#">LDAPTrustedClientCert type directory-path/filename/nickname [password]</a>	
Sets the file containing or nickname referring to a per connection client certificate. Not all LL support per connection client certificates.	
<a href="#">LDAPTrustedGlobalCert type directory-path/filename [password]</a>	
Sets the file or database containing global trusted Certificate Authority or global client certifi	
<a href="#">LDAPTrustedMode type</a>	

Specifies the SSL/TLS mode to be used when connecting to an LDAP server.	
<a href="#">LDAPVerifyServerCert</a> <i>On Off</i>	On
Force server certificate verification	
<a href="#">&lt;Limit method [method] ... &gt; ... &lt;/Limit&gt;</a>	
Restrict enclosed access controls to only certain HTTP methods	
<a href="#">&lt;LimitExcept method [method] ... &gt; ... &lt;/LimitExcept&gt;</a>	
Restrict access controls to all HTTP methods except the named ones	
<a href="#">LimitInternalRecursion</a> <i>number [number]</i>	10
Determine maximum number of internal redirects and nested subrequests	
<a href="#">LimitRequestBody</a> <i>bytes</i>	0
Restricts the total size of the HTTP request body sent from the client	
<a href="#">LimitRequestFields</a> <i>number</i>	100
Limits the number of HTTP request header fields that will be accepted from the client	
<a href="#">LimitRequestFieldSize</a> <i>bytes</i>	8190
Limits the size of the HTTP request header allowed from the client	
<a href="#">LimitRequestLine</a> <i>bytes</i>	8190
Limit the size of the HTTP request line that will be accepted from the client	
<a href="#">LimitXMLRequestBody</a> <i>bytes</i>	1000000
Limits the size of an XML-based request body	
<a href="#">Listen</a> [ <i>IP-address:</i> ] <i>portnumber [protocol]</i>	
IP addresses and ports that the server listens to	
<a href="#">ListenBacklog</a> <i>backlog</i>	
Maximum length of the queue of pending connections	
<a href="#">LoadFile</a> <i>filename [filename] ...</i>	
<a href="#">LoadModule</a> <i>module filename</i>	
<a href="#">&lt;Location URL-path URL&gt; ... &lt;/Location&gt;</a>	
Applies the enclosed directives only to matching URLs	
<a href="#">&lt;LocationMatch regex&gt; ... &lt;/LocationMatch&gt;</a>	
Applies the enclosed directives only to regular-expression matching URLs	
<a href="#">LockFile</a> <i>filename</i>	logs/accept.lock
Location of the accept serialization lock file	
<a href="#">LogFormat</a> <i>format nickname [nickname]</i>	"%h %l %u %t \"%r\" -

<a href="#"><u>LogLevel</u></a> <i>level</i>	warn
Controls the verbosity of the ErrorLog	
<a href="#"><u>MaxClients</u></a> <i>number</i>	
Maximum number of connections that will be processed simultaneously	
<a href="#"><u>MaxKeepAliveRequests</u></a> <i>number</i>	100
Number of requests allowed on a persistent connection	
<a href="#"><u>MaxMemFree</u></a> <i>KBytes</i>	0
Maximum amount of memory that the main allocator is allowed to hold without calling free	
<a href="#"><u>MaxRanges</u></a> <i>default   unlimited   none   number-of-ranges</i>	200
Number of ranges allowed before returning the complete resource	
<a href="#"><u>MaxRequestsPerChild</u></a> <i>number</i>	10000
Limit on the number of requests that an individual child server will handle during its life	
<a href="#"><u>MaxRequestsPerThread</u></a> <i>number</i>	0
<a href="#"><u>MaxSpareServers</u></a> <i>number</i>	10
Maximum number of idle child server processes	
<a href="#"><u>MaxSpareThreads</u></a> <i>number</i>	
Maximum number of idle threads	
<a href="#"><u>MaxThreads</u></a> <i>number</i>	2048
Set the maximum number of worker threads	
<a href="#"><u>MCacheMaxObjectCount</u></a> <i>value</i>	1009
<a href="#"><u>MCacheMaxObjectSize</u></a> <i>bytes</i>	10000
()	
<a href="#"><u>MCacheMaxStreamingBuffer</u></a> <i>size in bytes</i>	100000 MCacheMaxOb +
<a href="#"><u>MCacheMinObjectSize</u></a> <i>bytes</i>	0
()	
<a href="#"><u>MCacheRemovalAlgorithm</u></a> <i>LRU GDSF</i>	GDSF
<a href="#"><u>MCacheSize</u></a> <i>KBytes</i>	100
(KByte )	

<a href="#">MergeTrailers [on off]</a>	off
Determines whether trailers are merged into headers	
<a href="#">MetaDir <i>directory</i></a>	.web
CERN	
<a href="#">MetaFiles on off</a>	off
CERN	
<a href="#">MetaSuffix <i>suffix</i></a>	.meta
CERN	
<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>	5
Minimum number of idle child server processes	
<a href="#">MinSpareThreads <i>number</i></a>	
Minimum number of idle threads available to handle request spikes	
<a href="#">MMapFile <i>file-path [file-path] ...</i></a>	
<a href="#">ModMimeUsePathInfo On Off</a>	Off
Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename	
<a href="#">MultiviewsMatch</a>	NegotiatedOnly
<a href="#">Any NegotiatedOnly Filters Handlers</a>	
<a href="#">[Handlers Filters]</a>	
The types of files that will be included when searching for a matching file with MultiViews	
<a href="#">NameVirtualHost <i>addr[:port]</i></a>	
Designates an IP address for name-virtual hosting	
<a href="#">NoProxy <i>host [host] ...</i></a>	
Hosts, domains, or networks that will be connected to directly	
<a href="#">NWSSLTrustedCerts <i>filename [filename] ...</i></a>	
List of additional client certificates	
<a href="#">NWSSLUpgradeable [<i>IP-address:</i>]portnumber</a>	
Allows a connection to be upgraded to an SSL connection upon request	
<a href="#">Options [+ -]option [[+ -]option] ...</a>	All
Configures what features are available in a particular directory	
<a href="#">Order <i>ordering</i></a>	Deny,Allow
Allow Deny .	
<a href="#">PassEnv <i>env-variable [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="#">Protocol <i>protocol</i></a>	
Protocol for a listening socket	
<a href="#">ProtocolEcho On Off</a>	
echo	
<a href="#">&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</a>	
Container for directives applied to proxied resources	
<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="#">ProxyFtpDirCharset <i>character set</i></a>	ISO-8859-1
Define the character set for proxied FTP listings	
<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</i>] ! <i>url</i> [<i>key=value</i> [<i>key=value</i> ...]] [<i>nocanon</i>] [<i>interpolate</i>]</a>	
Maps remote servers into the local server URL-space	
<a href="#">ProxyPassInterpolateEnv On Off</a>	Off
Enable Environment Variable interpolation in Reverse Proxy configurations	
<a href="#">ProxyPassMatch [<i>regex</i>] ! <i>url</i> [<i>key=value</i> [<i>key=value</i> ...]]</a>	
Maps remote servers into the local server URL-space using regular expressions	
<a href="#">ProxyPassReverse [<i>path</i>] <i>url</i> [<i>interpolate</i>]</a>	

Adjusts the URL in HTTP response headers sent from a reverse proxied server	
<a href="#">ProxyPassReverseCookieDomain</a> <i>internal-domain public-domain [interpolate]</i>	
Adjusts the Domain string in Set-Cookie headers from a reverse- proxied server	
<a href="#">ProxyPassReverseCookiePath</a> <i>internal-path public-path [interpolate]</i>	
Adjusts the Path string in Set-Cookie headers from a reverse- proxied server	
<a href="#">ProxyPreserveHost</a> On Off	Off
Use incoming Host HTTP request header for proxy request	
<a href="#">ProxyReceiveBufferSize</a> <i>bytes</i>	0
Network buffer size for proxied HTTP and FTP connections	
<a href="#">ProxyRemote</a> <i>match remote-server</i>	
Remote proxy used to handle certain requests	
<a href="#">ProxyRemoteMatch</a> <i>regex remote-server</i>	
Remote proxy used to handle requests matched by regular expressions	
<a href="#">ProxyRequests</a> On Off	Off
Enables forward (standard) proxy requests	
<a href="#">ProxySCGIInternalRedirect</a> On Off	On
Enable or disable internal redirect responses from the backend	
<a href="#">ProxySCGISendfile</a> On Off  <i>Headername</i>	Off
Enable evaluation of X- <i>Sendfile</i> pseudo response header	
<a href="#">ProxySet</a> <i>url key=value [key=value ...]</i>	
Set various Proxy balancer or member parameters	
<a href="#">ProxyStatus</a> Off On Full	Off
Show Proxy LoadBalancer status in mod_status	
<a href="#">ProxyTimeout</a> <i>seconds</i>	
Network timeout for proxied requests	
<a href="#">ProxyVia</a> On Off Full Block	Off
Information provided in the Via HTTP response header for proxied requests	
<a href="#">ReadmeName</a> <i>filename</i>	
<a href="#">ReceiveBufferSize</a> <i>bytes</i>	0
TCP receive buffer size	
<a href="#">Redirect</a> [ <i>status</i> ] <i>URL-path URL</i>	
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="#">RegisterHttpMethod</a> <i>method [method [...]]</i>	
Register non-standard HTTP methods	
<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>	
Removes any output filter associations for a set of file extensions	
<a href="#">RemoveType</a> <i>extension [extension] ...</i>	
Removes any content type associations for a set of file extensions	
<a href="#">RequestHeader</a> <i>set append add unset header [value] [early env=[!]variable]</i>	
HTTP	
<a href="#">RequestReadTimeout</a> <i>[header=timeout[[-maxtimeout],MinRate=rate] [body=timeout[[-maxtimeout],MinRate=rate]</i>	
Set timeout values for receiving request headers and body from client.	
<a href="#">Require</a> <i>entity-name [entity-name] ...</i>	
Selects which authenticated users can access a resource	
<a href="#">RewriteBase</a> <i>URL-path</i>	
Sets the base URL for per-directory rewrites	
<a href="#">RewriteCond</a> <i>TestString CondPattern</i>	
Defines a condition under which rewriting will take place	

<a href="#">RewriteEngine on off</a>	off
Enables or disables runtime rewriting engine	
<a href="#">RewriteLock file-path</a>	
Sets the name of the lock file used for <a href="#">RewriteMap</a> synchronization	
<a href="#">RewriteLog file-path pipe</a>	
Sets the name of the file used for logging rewrite engine processing	
<a href="#">RewriteLogLevel Level</a>	0
Sets the verbosity of the log file used by the rewrite engine	
<a href="#">RewriteMap MapName MapType:MapSource</a>	
Defines a mapping function for key-lookup	
<a href="#">RewriteOptions Options</a>	
Sets some special options for the rewrite engine	
<a href="#">RewriteRule Pattern Substitution [flags]</a>	
Defines rules for the rewriting engine	
<a href="#">RLimitCPU seconds max [seconds max]</a>	
Limits the CPU consumption of processes launched by Apache children	
<a href="#">RLimitMEM bytes max [bytes max]</a>	
Limits the memory consumption of processes launched by Apache children	
<a href="#">RLimitNPROC number max [number max]</a>	
Limits the number of processes that can be launched by processes launched by Apache ch	
<a href="#">Satisfy Any All</a>	All
Interaction between host-level access control and user authentication	
<a href="#">ScoreBoardFile file-path</a>	logs/apache_runtime .
Location of the file used to store coordination data for the child processes	
<a href="#">Script method cgi-script</a>	
CGI .	
<a href="#">ScriptAlias URL-path file-path directory-path</a>	
URL CGI	
<a href="#">ScriptAliasMatch regex file-path directory-path</a>	
URL CGI	
<a href="#">ScriptInterpreterSource Registry Registry-Strict Script</a>	Script
Technique for locating the interpreter for CGI scripts	
<a href="#">ScriptLog file-path</a>	
CGI	

<a href="#">ScriptLogBuffer <i>bytes</i></a> PUT POST	1024
<a href="#">ScriptLogLength <i>bytes</i></a> CGI	10385760
<a href="#">ScriptSock <i>file-path</i></a> cgi	logs/cgisock
<a href="#">SecureListen [<i>IP-address:</i>]<i>portnumber</i></a> <a href="#">Certificate-Name [MUTUAL]</a> Enables SSL encryption for the specified port	
<a href="#">SeeRequestTail On Off</a> Determine if mod_status displays the first 63 characters of a request or the last 63, assuming itself is greater than 63 chars.	Off
<a href="#">SendBufferSize <i>bytes</i></a> TCP buffer size	0
<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="#">ServerLimit <i>number</i></a> Upper limit on configurable number of processes	
<a href="#">ServerName [<i>scheme://</i>]<i>fully-qualified-domain-name[:port]</i></a> Hostname and port that the server uses to identify itself	
<a href="#">ServerPath <i>URL-path</i></a> Legacy URL pathname for a name-based virtual host that is accessed by an incompatible b	
<a href="#">ServerRoot <i>directory-path</i></a> Base directory for the server installation	/usr/local/apache
<a href="#">ServerSignature On Off EMail</a> Configures the footer on server-generated documents	Off
<a href="#">ServerTokens</a> <a href="#">Major Minor Min[imal]  Prod[uctOnly]  OS Full</a> Configures the Server HTTP response header	Full
<a href="#">SetEnv <i>env-variable value</i></a>	
<a href="#">SetEnvIf <i>attribute regex [!]<i>env-variable</i>[=<i>value</i>]</i></a>	

<a href="#">[!]env-variable[=value]] ...</a>	
<a href="#">SetEnvIfNoCase attribute regex [!]env-variable[=value] [!]env-variable[=value]] ...</a>	
<a href="#">SetHandler handler-name None</a>	
Forces all matching files to be processed by a handler	
<a href="#">SetInputFilter filter[:filter...]</a>	
Sets the filters that will process client requests and POST input	
<a href="#">SetOutputFilter filter[:filter...]</a>	
Sets the filters that will process responses from the server	
<a href="#">SSIEnableAccess on off</a>	off
Enable the -A flag during conditional flow control processing.	
<a href="#">SSIEndTag tag</a>	"-->"
String that ends an include element	
<a href="#">SSLErrorMsg message</a>	"[an error occurred +
Error message displayed when there is an SSI error	
<a href="#">SSIETag on off</a>	off
Controls whether ETags are generated by the server.	
<a href="#">SSILastModified on off</a>	off
Controls whether Last -Modified headers are generated by the server.	
<a href="#">SSIStartTag tag</a>	"<!--#"
String that starts an include element	
<a href="#">SSITimeFormat formatstring</a>	"%A, %d-%b-%Y %H:%M +
Configures the format in which date strings are displayed	
<a href="#">SSIUndefinedEcho string</a>	"(none)"
String displayed when an unset variable is echoed	
<a href="#">SSLCACertificateFile file-path</a>	
File of concatenated PEM-encoded CA Certificates for Client Auth	
<a href="#">SSLCACertificatePath directory-path</a>	
Directory of PEM-encoded CA Certificates for Client Auth	
<a href="#">SSLCADNRequestFile file-path</a>	
File of concatenated PEM-encoded CA Certificates for defining acceptable CA names	
<a href="#">SSLCADNRequestPath directory-path</a>	

Directory of PEM-encoded CA Certificates for defining acceptable CA names	
<a href="#"><u>SSLCARevocationFile</u></a> <i>file-path</i>	
File of concatenated PEM-encoded CA CRLs for Client Auth	
<a href="#"><u>SSLCARevocationPath</u></a> <i>directory-path</i>	
Directory of PEM-encoded CA CRLs for Client Auth	
<a href="#"><u>SSLCertificateChainFile</u></a> <i>file-path</i>	
File of PEM-encoded Server CA Certificates	
<a href="#"><u>SSLCertificateFile</u></a> <i>file-path</i>	
Server PEM-encoded X.509 Certificate 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>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL handshake	
<a href="#"><u>SSLCompression</u></a> <i>on off</i>	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> <i>on off optional</i>	off
SSL Engine Operation Switch	
<a href="#"><u>SSLFIPS</u></a> <i>on off</i>	off
SSL FIPS mode Switch	
<a href="#"><u>SSLHonorCipherOrder</u></a> <i>flag</i>	
Option to prefer the server's cipher preference order	
<a href="#"><u>SSLInsecureRenegotiation</u></a> <i>flag</i>	off
Option to enable support for insecure renegotiation	
<a href="#"><u>SSLMutex</u></a> <i>type</i>	none
Semaphore for internal mutual exclusion of operations	
<a href="#"><u>SSLOptions</u></a> <i>[+ -]option ...</i>	
Configure various SSL engine run-time options	
<a href="#"><u>SSLPassPhraseDialog</u></a> <i>type</i>	builtin
Type of pass phrase dialog for encrypted private keys	
<a href="#"><u>SSLProtocol</u></a> <i>[+ -]protocol ...</i>	all
Configure usable SSL protocol flavors	
<a href="#"><u>SSLProxyCACertificateFile</u></a> <i>file-path</i>	

<a href="#">File of concatenated PEM-encoded CA Certificates for Remote Server Auth</a>	
<a href="#">SSLProxyCACertificatePath <i>directory-path</i></a>	
Directory of PEM-encoded CA Certificates for Remote Server Auth	
<a href="#">SSLProxyCARevocationFile <i>file-path</i></a>	
File of concatenated PEM-encoded CA CRLs for Remote Server Auth	
<a href="#">SSLProxyCARevocationPath <i>directory-path</i></a>	
Directory of PEM-encoded CA CRLs for Remote Server Auth	
<a href="#">SSLProxyCheckPeerCN <i>on off</i></a>	off
Whether to check the remote server certificates CN field	
<a href="#">SSLProxyCheckPeerExpire <i>on off</i></a>	off
Whether to check if remote server certificate is expired	
<a href="#">SSLProxyCipherSuite <i>cipher-spec</i></a>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL proxy handshake	
<a href="#">SSLProxyEngine <i>on off</i></a>	off
SSL Proxy Engine Operation Switch	
<a href="#">SSLProxyMachineCertificateChainFile <i>filename</i></a>	
File of concatenated PEM-encoded CA certificates to be used by the proxy for choosing a c	
<a href="#">SSLProxyMachineCertificateFile <i>filename</i></a>	
File of concatenated PEM-encoded client certificates and keys to be used by the proxy	
<a href="#">SSLProxyMachineCertificatePath <i>directory</i></a>	
Directory of PEM-encoded client certificates and keys to be used by the proxy	
<a href="#">SSLProxyProtocol <i>[+ -]protocol ...</i></a>	all
Configure usable SSL protocol flavors for proxy usage	
<a href="#">SSLProxyVerify <i>level</i></a>	none
Type of remote server Certificate verification	
<a href="#">SSLProxyVerifyDepth <i>number</i></a>	1
Maximum depth of CA Certificates in Remote Server Certificate verification	
<a href="#">SSLRandomSeed <i>context source [bytes]</i></a>	
Pseudo Random Number Generator (PRNG) seeding source	
<a href="#">SSLRenegBufferSize <i>bytes</i></a>	131072
Set the size for the SSL renegotiation buffer	
<a href="#">SSLRequire <i>expression</i></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 type</u></a>	none
Type of the global/inter-process SSL Session Cache	
<a href="#"><u>SSLSessionCacheTimeout seconds</u></a>	300
Number of seconds before an SSL session expires in the Session Cache	
<a href="#"><u>SSLSessionTicketKeyFile file-path</u></a>	
Persistent encryption/decryption key for TLS session tickets	
<a href="#"><u>SSLSessionTickets on off</u></a>	on
Enable or disable use of TLS session tickets	
<a href="#"><u>SSLStrictSNIVHostCheck on off</u></a>	off
Whether to allow non SNI clients to access a name based virtual host.	
<a href="#"><u>SSLUserName varname</u></a>	
Variable name to determine user name	
<a href="#"><u>SSLVerifyClient level</u></a>	none
Type of Client Certificate verification	
<a href="#"><u>SSLVerifyDepth number</u></a>	1
Maximum depth of CA Certificates in Client Certificate verification	
<a href="#"><u>StartServers number</u></a>	
Number of child server processes created at startup	
<a href="#"><u>StartThreads number</u></a>	
Number of threads created on startup	
<a href="#"><u>Substitute s/pattern/substitution/[infq]</u></a>	
Pattern to filter the response content	
<a href="#"><u>SubstituteInheritBefore on off</u></a>	off
Change the merge order of inherited patterns	
<a href="#"><u>Suexec On Off</u></a>	
Enable or disable the suEXEC feature	
<a href="#"><u>SuexecUserGroup User Group</u></a>	
CGI	
<a href="#"><u>ThreadLimit number</u></a>	
Sets the upper limit on the configurable number of threads per child process	
<a href="#"><u>ThreadsPerChild number</u></a>	
Number of threads created by each child process	
<a href="#"><u>ThreadStackSize size</u></a>	

The size in bytes of the stack used by threads handling client connections	
<a href="#">TimeOut <i>seconds</i></a>	300
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 behaviour 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="#">UnsetEnv <i>env-variable [env-variable] ...</i></a>	
<a href="#">UseCanonicalName <i>On Off DNS</i></a>	Off
Configures how the server determines its own name and port	
<a href="#">UseCanonicalPhysicalPort <i>On Off</i></a>	Off
Configures how the server determines its own name and 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="#">VirtualDocumentRoot <i>interpolated-directory none</i></a>	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">VirtualDocumentRootIP <i>interpolated-directory none</i></a>	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">&lt;VirtualHost <i>addr[:port] [addr[:port]] ...&gt; ...</i></a> <a href="#">&lt;/VirtualHost&gt;</a>	
Contains directives that apply only to a specific hostname or IP address	
<a href="#">VirtualScriptAlias <i>interpolated-directory none</i></a>	none
Dynamically configure the location of the CGI directory for a given virtual host	
<a href="#">VirtualScriptAliasIP <i>interpolated-directory none</i></a>	none
Dynamically configure the location of the cgi directory for a given virtual host	
<a href="#">Win32DisableAcceptEx</a>	
Use accept() rather than AcceptEx() to accept network connections	

[XBitHack on|off|full](#)

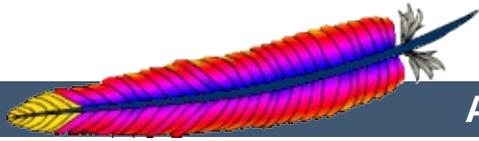
| off

Parse SSI directives in files with the execute bit set

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



(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)

## [beos](#)

BeOS .

## [event](#)

An experimental variant of the standard [worker](#) MPM

## [mpm\\_netware](#)

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](#)

This Multi-Processing Module is optimized for Windows NT.

## [worker](#)

Multi-Processing Module implementing a hybrid multi-threaded multi-process web server



A | C | D | E | F | H | I | L | M | N | P | R | S | U |  
V

### [mod actions](#)

CGI .

### [mod alias](#)

,

URL

### [mod asis](#)

HTTP

### [mod auth basic](#)

Basic authentication

### [mod auth digest](#)

MD5 Digest Authentication .

### [mod authn alias](#)

Provides the ability to create extended authentication providers based on actual providers

### [mod authn anon](#)

"(anonymous)"

### [mod authn dbd](#)

User authentication using an SQL database

### [mod authn dbm](#)

DBM

### [mod authn default](#)

### [mod authn file](#)

### [mod authnz ldap](#)

Allows an LDAP directory to be used to store the database for HTTP Basic authentication.

[mod\\_authz\\_dbm](#)

DBM

[mod\\_authz\\_default](#)

[mod\\_authz\\_groupfile](#)

[mod\\_authz\\_host](#)

( IP )

[mod\\_authz\\_owner](#)

[mod\\_authz\\_user](#)

[mod\\_autoindex](#)

ls Win32 dir

[mod\\_cache](#)

URI .

[mod\\_cern\\_meta](#)

CERN

[mod\\_cgi](#)

CGI

[mod\\_cgid](#)

CGI CGI

[mod\\_charset\\_lite](#)

[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\\_dir](#)

" " index

### [mod\\_disk\\_cache](#)

Content cache storage manager keyed to URIs

### [mod\\_dumpio](#)

Dumps all I/O to error log as desired.

### [mod\\_echo](#)

echo

### [mod\\_env](#)

CGI SSI

### [mod\\_example](#)

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\\_ident](#)

RFC 1413 ident

### [mod\\_imagemap](#)

(imagemap)

### [mod\\_include](#)

Server-parsed html documents (Server Side Includes)

### [mod\\_info](#)

### [mod\\_isapi](#)

Windows ISAPI Extension

### [mod\\_ldap](#)

LDAP connection pooling and result caching services for use by other LDAP modules

### [mod\\_log\\_config](#)

### [mod\\_log\\_forensic](#)

Forensic Logging of the requests made to the server

### [mod\\_logio](#)

### [mod\\_mem\\_cache](#)

URI .

### [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\\_proxy](#)

HTTP/1.1 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\\_ftp](#)

FTP support module for [mod\\_proxy](#)

### [mod\\_proxy\\_http](#)

HTTP support module for [mod\\_proxy](#)

### [mod\\_proxy\\_scgi](#)

SCGI gateway module for [mod\\_proxy](#)

### [mod\\_reqtimeout](#)

Set timeout and minimum data rate for receiving requests

### [mod\\_rewrite](#)

Provides a rule-based rewriting engine to rewrite requested URLs on the fly

### [mod\\_setenvif](#)

### [mod\\_so](#)

### [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\\_userdir](#)

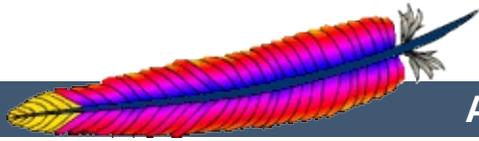
## [mod\\_usertrack](#)

*Clickstream* logging of user activity on a site

## [mod\\_version](#)

## [mod\\_vhost\\_alias](#)

Provides for dynamically configured mass virtual hosting



| | [FAQ](#) | |



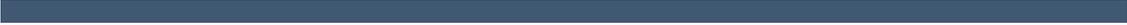
## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



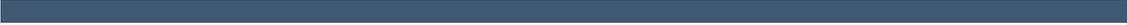
[Apache HTTP Server Version 2.2](#) .





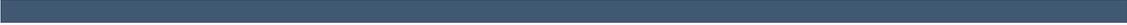
- [1.3 2.0](#)
- [2.0](#)
- 





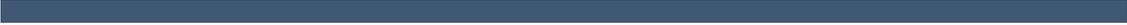
- [\[redacted\]](#)
- [\[redacted\]](#)
- [\[redacted\]](#)
- [\[redacted\]](#)
- [Directory, Location, Files](#)
- [\[redacted\]](#)
- [\[redacted\]](#)
- [URL](#)
- [\[redacted\]](#)
- [\(DSO\)](#)
- [\(content negotiation\)](#)
- [\[redacted\]](#)
- [\[redacted\]](#)
- [\(MPM\)](#)
- [\[redacted\]](#)
- [\[redacted\]](#)
- [\[redacted\]](#)
- [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](#)

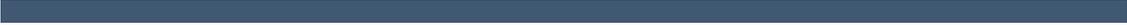
- [HPUX](#)

- [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 \[redacted\]](#)
- [MPM beos](#)
- [MPM event](#)
- [MPM netware](#)
- [MPM os2](#)
- [MPM prefork](#)
- [MPM winnt](#)
- [MPM worker](#)

- [mod\\_actions](#)
- [mod\\_alias](#)
- [mod\\_asis](#)
- [mod\\_auth\\_basic](#)
- [mod\\_auth\\_digest](#)
- [mod\\_authn\\_alias](#)
- [mod\\_authn\\_anon](#)
- [mod\\_authn\\_dbd](#)
- [mod\\_authn\\_dbm](#)
- [mod\\_authn\\_default](#)
- [mod\\_authn\\_file](#)
- [mod\\_authnz\\_ldap](#)
- [mod\\_authz\\_dbm](#)
- [mod\\_authz\\_default](#)
- [mod\\_authz\\_groupfile](#)
- [mod\\_authz\\_host](#)
- [mod\\_authz\\_owner](#)
- [mod\\_authz\\_user](#)
- [mod\\_autoindex](#)
- [mod\\_cache](#)

- [mod\\_cern\\_meta](#)
- [mod\\_cgi](#)
- [mod\\_cgid](#)
- [mod\\_charset\\_lite](#)
- [mod\\_dav](#)
- [mod\\_dav\\_fs](#)
- [mod\\_dav\\_lock](#)
- [mod\\_dbd](#)
- [mod\\_deflate](#)
- [mod\\_dir](#)
- [mod\\_disk\\_cache](#)
- [mod\\_dumpio](#)
- [mod\\_echo](#)
- [mod\\_env](#)
- [mod\\_example](#)
- [mod\\_expires](#)
- [mod\\_ext\\_filter](#)
- [mod\\_file\\_cache](#)
- [mod\\_filter](#)
- [mod\\_headers](#)
- [mod\\_ident](#)
- [mod\\_imagemap](#)
- [mod\\_include](#)
- [mod\\_info](#)
- [mod\\_isapi](#)
- [mod\\_ldap](#)
- [mod\\_log\\_config](#)
- [mod\\_log\\_forensic](#)
- [mod\\_logio](#)
- [mod\\_mem\\_cache](#)
- [mod\\_mime](#)
- [mod\\_mime\\_magic](#)
- [mod\\_negotiation](#)
- [mod\\_nw\\_ssl](#)

- [mod\\_proxy](#)
- [mod\\_proxy\\_ajp](#)
- [mod\\_proxy\\_balancer](#)
- [mod\\_proxy\\_connect](#)
- [mod\\_proxy\\_ftp](#)
- [mod\\_proxy\\_http](#)
- [mod\\_proxy\\_scgi](#)
- [mod\\_reqtimeout](#)
- [mod\\_rewrite](#)
- [mod\\_setenvif](#)
- [mod\\_so](#)
- [mod\\_speling](#)
- [mod\\_ssl](#)
- [mod\\_status](#)
- [mod\\_substitute](#)
- [mod\\_suexec](#)
- [mod\\_unique\\_id](#)
- [mod\\_userdir](#)
- [mod\\_usertrack](#)
- [mod\\_version](#)
- [mod\\_vhost\\_alias](#)



---

- 

- [Apache API](#)

- [APR](#)

- [Apache 2.0](#)

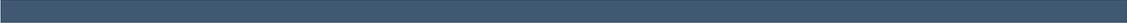
- [Apache 2.0 \(hook\)](#)

- [Apache 1.3 Apache 2.0](#)

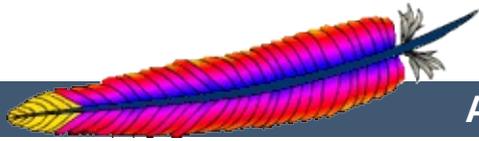
- [Apache 2.0](#)

- [Apache 2.0](#)





- 
- 
- 
- 



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



.



---

[httpd](#)

[apachectl](#)

[ab](#)

[apxs](#)

(APache eXtenSion tool)

[configure](#)

[dbmmanage](#)

basic authentication DBM

[htcacheclean](#)

[htdigest](#)

digest authentication

[htpasswd](#)

basic authentication

[logresolve](#)

IP-

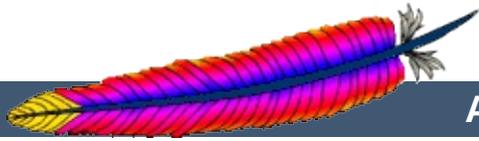
[rotatelog](#)

[suexec](#)

(Switch User For Exec)

[manpage](#)

manpage .



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

# 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. The module and this documentation are based on Ralf S. Engelschall's `mod_ssl` project.



- 
- [Introduction](#)
  - [Compatibility](#)
  - [How-To](#)
  - [Frequently Asked Questions](#)
  - [Glossary](#)

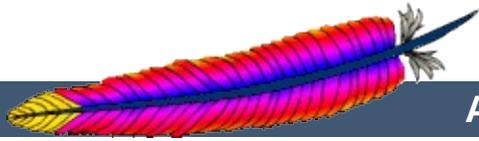


Extensive documentation on the directives and environment variables provided by this module is provided in the [mod\\_ssl reference documentation](#).

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

# 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 and tricks](#)
- [When \*\*NOT\*\* to use mod\\_rewrite](#)
- [RewriteRule Flags](#)
- [Technical details](#)

## See also

[mod\\_rewrite reference documentation](#)

[Mapping URLs to the Filesystem](#)

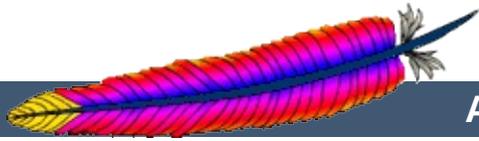
[mod\\_rewrite wiki](#)

[Glossary](#)

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



(Virtual Host) ( , www.company1.co  
 www.company2.com) . IP  
 based)" IP " (name-based) " .

IP . 1.1 IP  
 (host-based) IP (non-IP virtual hosts) .

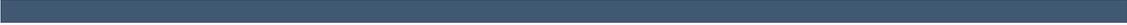
1.3

mod vhost alias

```

  IP
  
```





- (IP )
- IP ( IP )
- 
- (file descriptor)   (, )
- 
- 

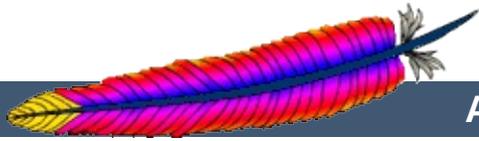


- [<VirtualHost>](#)
- [NameVirtualHost](#)
- [ServerName](#)
- [ServerAlias](#)
- [ServerPath](#)

-S ., :

```
/usr/local/apache2/bin/httpd -S
```

( [httpd](#) .) . IP



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

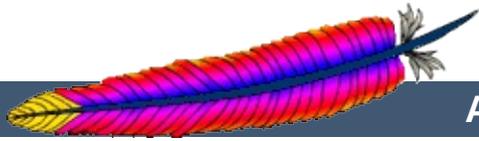
# Frequently Asked Questions

The FAQ has 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

## Developer Documentation for Apache 2.x

### Warning

Many of the documents listed here are in need of update. They are in different stages of progress. Please be patient, and point out any discrepancies or errors on the developer/ pages directly to the [dev@httpd.apache.org](mailto:dev@httpd.apache.org) mailing list.



- [Apache 1.3 API Notes](#)
- [Apache 2.0 Hook Functions](#)
- [Request Processing in Apache 2.0](#)
- [How filters work in Apache 2.0](#)
- [Converting Modules from Apache 1.3 to Apache 2.0](#)
- [Documenting Apache 2.0](#)
- [Apache 2.x Thread Safety Issues](#)

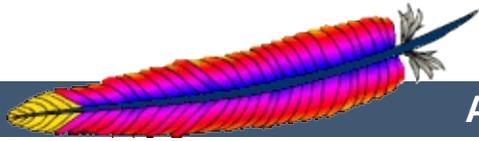


- 
- Developer articles at [apachetutor](#) include:
    - [Request Processing in Apache](#)
    - [Configuration for Modules](#)
    - [Resource Management in Apache](#)
    - [Connection Pooling in Apache](#)
    - [Introduction to Buckets and Brigades](#)

---

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

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



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

.

2.1 . ,

(,)  
)

""

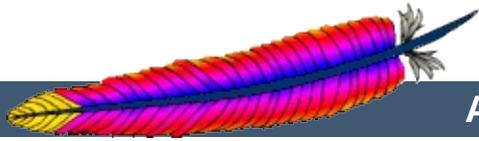
".

URL

mod\_rewrite .

mod\_rewrite .

.



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

## 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\_cerro

IP .

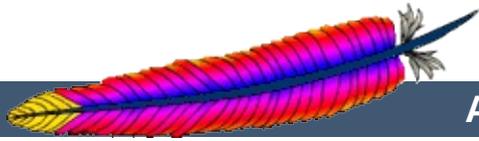
httpd -t tes

## MPM

MPM ThreadStackSize .

ap\_register\_output\_filter\_protocol

ap\_filter\_protocol [mod\\_](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)



Microsoft Windows 2.0 , , .

: [Microsoft Windows](#)

.

: [Microsoft Windows](#)



---

## Novell NetWare

Novell NetWare 5.1 2.0

: [Novell NetWare](#)

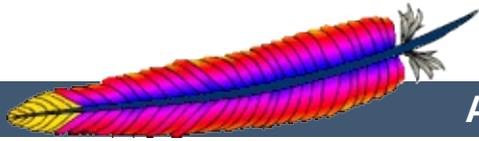
## EBCDIC

1.3 EBCDIC

(-ASCII)

: 2.0 . ,  
.

: [EBCDIC](#)



| | [FAQ](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#)

## How-To /





(authentication) . (authorization)

.  
:  
: [.htaccess](#)

**CGI**

CGI (Common Gateway Interface) CGI CGI  
, ( ) .  
. CGI , CGI .  
:  
: [CGI:](#)

**.htaccess**

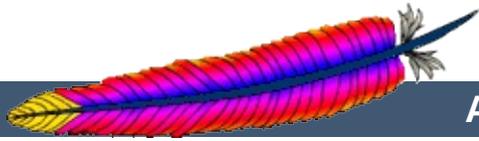
.htaccess .  
, .  
:  
: [.htaccess](#)

**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](#) | |



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Programs](#)

## suexec -

CGI suexec .  
root suexec setuid  
root .

suexec suexec

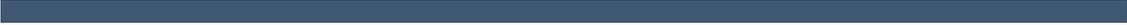
<http://httpd.apache.org/docs/2.2/suexec.html> .



---

**suexec -V**

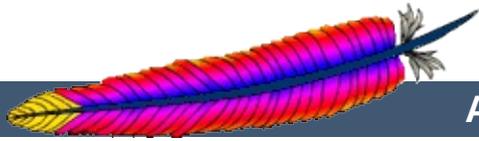




**-v**

root suexec .

.



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 is [mod\\_authz\\_host](#). Other modules discussed in this document include [mod\\_setenvif](#) and [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 `Allow` and `Deny` directives let you allow and deny access based on the host name, or host address, of the machine requesting a document. The `Order` directive goes hand-in-hand with these two, and tells Apache in which order to apply the filters.

The usage of these directives is:

```
Allow from address
```

where *address* is an IP address (or a partial IP address) or a fully qualified domain name (or a partial domain name); you may provide multiple addresses or domain names, if desired.

For example, if you have someone spamming your message board, and you want to keep them out, you could do the following:

```
Deny from 10.252.46.165
```

Visitors coming from that address 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.

```
Deny from 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:

```
Deny from 192.168.205  
Deny from phishers.example.com moreidiots.example  
Deny from ke
```

Using [Order](#) will let you be sure that you are actually restricting things to the group that you want to let in, by combining a [Deny](#) and an [Allow](#) directive:

```
Order deny,allow  
Deny from all  
Allow from dev.example.com
```

Listing just the [Allow](#) directive would not do what you want, because it will let folks from that host in, in addition to letting everyone in. What you want is to let *only* those folks in.



**Access Control by Environment Variable**

`mod_authz_host`, in conjunction with `mod_setenvif`, can be used to restrict access to your website based on the value of arbitrary environment variables. This is done with the `Allow from env=` and `Deny from env=` syntax.

```
SetEnvIf User-Agent BadBot GoAway=1
Order allow,deny
Allow from all
Deny from env=GoAway
```

**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.

In the above example, the environment variable `GoAway` is set to `1` if the `User-Agent` matches the string `BadBot`. Then we deny access for any request when this variable is set. This blocks that particular user agent from the site.

An environment variable test can be negated using the `=!` syntax:

```
Allow from env=!GoAway
```



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 6am, 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.



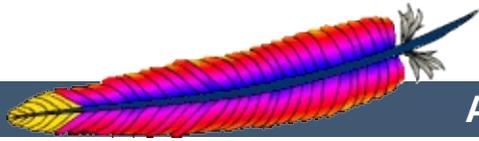
You should also read the documentation for [mod\\_auth\\_basic](#) and [mod\\_authz\\_host](#) which contain some more information about how this all works. [mod\\_authn\\_alias](#) can also help in simplifying certain authentication configurations.

See the [Authentication and Authorization](#) howto.

---

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

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Miscellaneous Documentation](#)

## Password Formats

Notes about the password encryption formats generated and understood by Apache.



There are four formats that Apache recognizes for basic-authentication passwords. Note that not all formats work on every platform:

### **PLAIN TEXT (i.e. *unencrypted*)**

Windows, BEOS, & Netware only.

### **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.

### **SHA1**

"{SHA}" + Base64-encoded SHA-1 digest of the password.

### **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.

## **Generating values with htpasswd**

### **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.getInst
```

### 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`

### 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)
```

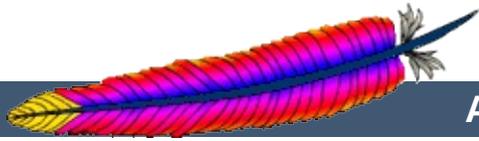
## PostgreSQL (with the contrib/pgcrypto functions installed)

```
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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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.

### 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.



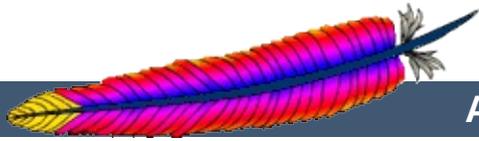
## Examples

```
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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 strings *Substitution* and *TestString* as outlined in the following chapters. 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.

```
RewriteCond %{DOCUMENT_ROOT}/$1 !-f
RewriteCond %{HTTP_HOST} ^(admin.example.com)$
RewriteRule ^/?([a-z]+)/(.*)$ /admin.foo?page=$1&id=$2&host=%1 [PT]
```

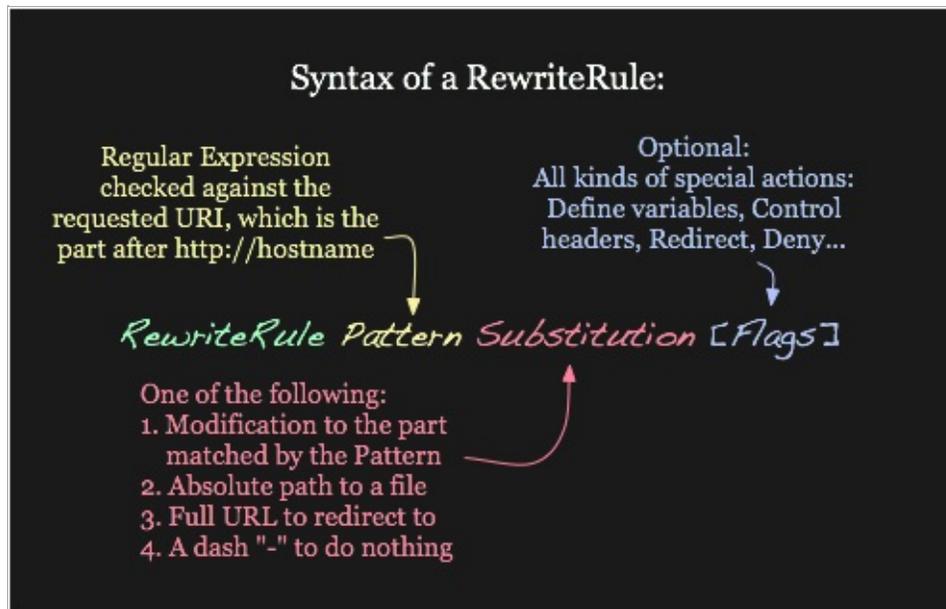
**Figure 1:** The back-reference flow through a rule.  
In this example, a request for `/test/1234` would be transformed into `/admin.foo?page=test&id=1234&host=admin.example.com`.



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/web
```

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://site2.example.com/seeproduct.html [R]
```

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/productdb/$1
```

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

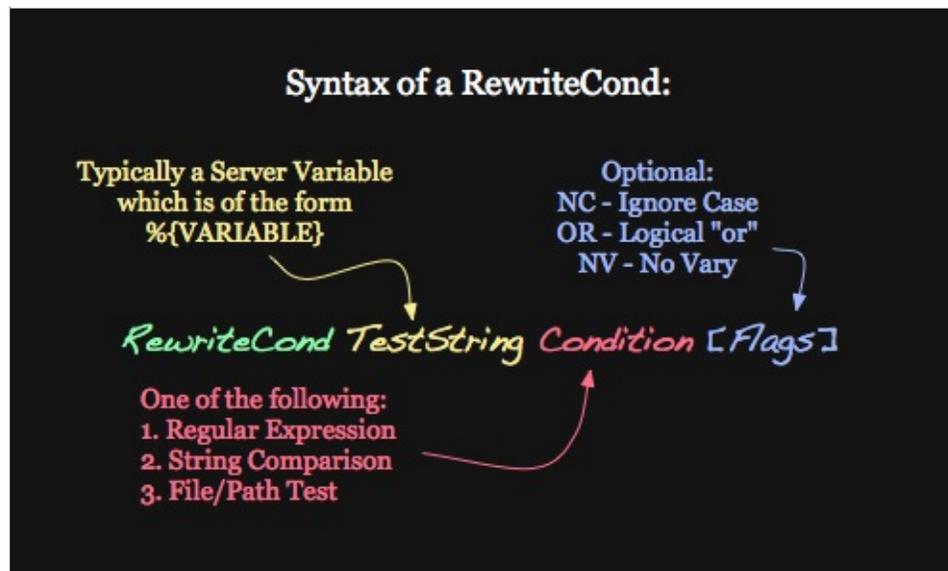
The behavior of a [RewriteRule](#) can be modified by the application of one or more flags to the end of the rule. For example, the matching behavior of a rule can be made case-insensitive by the application of the `[NC]` flag:

```
RewriteRule ^puppy.html smalldog.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.com$1
```

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 [RewriteCond](#)s 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](#).



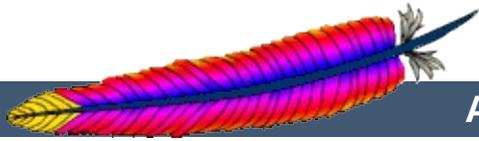
Rewriting is typically configured in the main server configuration setting (outside any `<Directory>` section) or inside `<VirtualHost>` containers. This is the easiest way to do rewriting and is recommended. It is possible, however, to do rewriting inside `<Directory>` sections or `.htaccess files` 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 `C0`, 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.

`mod_rewrite` has to unescape URLs before mapping them, so backreferences will be 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
```

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`.

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.



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, will be 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]
```

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. Prior to 2.2.28, a value of 0 indicates immediate expiration and it was not possible to specify session lifetime if any later parameters (Path, Secure, httponly) were 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 `HttP0nly`, `true`, or `1`, the cookie will have the `HttP0nly` flag set, which means that the cookie will be inaccessible to JavaScript code on browsers that support this feature.

Several examples are offered here:

```
RewriteEngine On
RewriteRule ^/index\.html -
[CO=frontdoor:yes:.example.com:1440:/]
```

In the example give, 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 will be set to expire in 1440 minutes (24 hours) and will be 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 will be 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:1]  
CustomLog logs/access_log combined env=!image
```

Note that this same effect can be obtained using [SetEnvIf](#). This technique is offered as an example, not as a recommendation.



## 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-php]
```

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=application/x-httpd-php-source]
```

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.

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,PT]
```



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).



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://images.example.com$1  
[P,NC]
```



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#$1 [NE,R]
```

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.



## Proxy

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://images.example.com/$1.$2 [P]
```

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.

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.gif [PT]
```

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 [QSA]
```

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.



## 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 3 lines if we
# meant to go to the "else" stanza.
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
```



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:

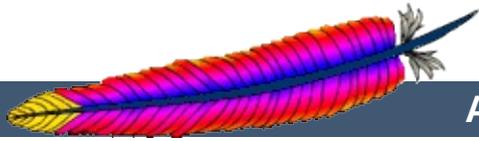
```
# Serve .pl files as plain text
RewriteRule \.pl$ - [T=text/plain]
```

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)
- [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.html [PT]
```



### 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 [R]
```

### 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.example.com/docs/$1
[R,L]
```

### With `RedirectMatch`

```
RedirectMatch ^/docs/(.*) http://new.example.com/docs/$1
```

### With `Redirect`

```
Redirect /docs/ http://new.example.com/docs/
```



### 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 [H=cgi-script]
```



## 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.php
# when and only when document.php exists
<Directory /var/www/htdocs>
  RewriteEngine on
  RewriteBase /var/www/htdocs

  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.example.com

  Redirect / http://www.example.com/
</VirtualHost>

<VirtualHost *:80>
  ServerName www.example.com
</VirtualHost>
```

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\.example\.com [NC]
RewriteCond %{HTTP_HOST} !^$
RewriteCond %{SERVER_PORT} !^80$
RewriteRule ^/?(.*) http://www.example.com:%
{SERVER_PORT}/$1 [L,R,NE]
```

And for a site running on port 80

```
RewriteCond %{HTTP_HOST} !^www\.example\.com [NC]
```

```
RewriteCond %{HTTP_HOST} !^$  
RewriteRule ^/?(.*) http://www.example.com/$1 [L,R,NE]
```

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.%{HTTP_HOST}/$1 [L,R,NE]
```

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}/dir1/%{REQUEST_URI} -f
RewriteRule ^(.+) %{DOCUMENT_ROOT}/dir1/$1 [L]

# second try to find it in dir2/...
# ...and if found stop and be happy:
RewriteCond %{DOCUMENT_ROOT}/dir2/%{REQUEST_URI} -f
RewriteRule ^(.+) %{DOCUMENT_ROOT}/dir2/$1 [L]

# else go on for other Alias or ScriptAlias directives,
# 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:/path/to/map.mirrors
RewriteCond %{REMOTE_HOST} ([a-z]+)$ [NC]
RewriteRule ^/(.*)$
${multiplex:%1|http://www.example.com/}$1 [R,L]
```

```
## 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} ^Mozilla/3.*
RewriteRule ^foo\.html$ foo.NS.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx/ [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/[12]
RewriteRule ^foo\.html$ foo.20.html [L]

RewriteRule ^foo\.html$ foo.32.html [L]
```



## Description:

On some webservers there is more than one URL for a resource. Usually there are canonical URLs (which are be 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)/(.*) /dogs/$2 [R]
```

## Discussion:

This should really be accomplished with `Redirect` or `RedirectMatch` directives:

```
RedirectMatch ^/(puppies|canines)/(.*) /dogs/$2
```



## 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/about/
```

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_FILENAME} !-f
  RewriteCond /var/www/my_blog/%{REQUEST_FILENAME} !-d
  RewriteRule ^ index.php [PT]
</Directory>
```

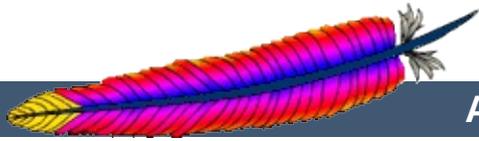
If, on the other hand, you wish to pass the requested URI as a query string argument to `index.php`, you can replace that `RewriteRule` with:

```
RewriteRule (.*) index.php?$1 [PT,QSA]
```

Note that these rulesets can be used in a `.htaccess` file, as well as in a `<Directory>` block.

Licensed under the [Apache License, Version 2.0](#).

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)
- [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.example.com [NC]  
RewriteRule \.(gif|jpg|png)$ - [F,NC]
```

In this second example, instead of failing the request, we display an alternate image instead.

```
RewriteCond %{HTTP_REFERER} !^$  
RewriteCond %{HTTP_REFERER} !www.example.com [NC]  
RewriteRule \.(gif|jpg|png)$ /images/go-away.png [R,NC]
```

In the third example, we redirect the request to an image on some other site.

```
RewriteCond %{HTTP_REFERER} !^$  
RewriteCond %{HTTP_REFERER} !www.example.com [NC]
```

```
RewriteRule \.(gif|jpg|png)$  
http://other.example.com/image.gif [R,NC]
```

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 localreferer  
<FilesMatch \.(jpg|png|gif)$>  
Order deny,allow  
Deny from all  
Allow from 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 client `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} ^NameOfBadRobot
RewriteCond %{REMOTE_ADDR} =123\.45\.67\.[8-9]
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 ^NameOfBadRobot goaway
<Location /secret/files>
Order allow,deny
```

```
Allow from all
Deny from env=goaway
</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/hosts.deny
RewriteCond ${hosts-deny:%{REMOTE_ADDR}|NOT-FOUND} !=NOT-FOUND [OR]
RewriteCond ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND} !=NOT-FOUND
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/deflector.map

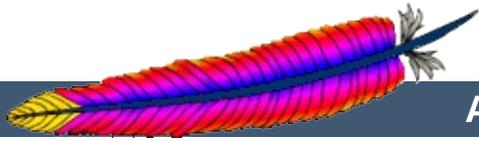
RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}} =-
RewriteRule ^ %{HTTP_REFERER} [R,L]

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}|NOT-FOUND} !=NOT-FOUND
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

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.html -
http://badguys.example.com/bad/index2.html -
http://badguys.example.com/bad/index3.html
http://somewhere.example.com/
```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)
- [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}} ^www\.[
([^.]+)\.example\.com$
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 %b" vcommon
CustomLog logs/access_log vcommon

<Directory /www/hosts>
    # ExecCGI is needed here because we can't force
    # CGI execution in the way that ScriptAlias does
    Options FollowSymLinks ExecCGI
</Directory>

RewriteEngine On

# a ServerName derived from a Host: header may be any case at
all
RewriteMap lowercase int:tolower

## deal with normal documents first:
# allow Alias /icons/ to work - repeat for other aliases
RewriteCond %{REQUEST_URI} !^/icons/
# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/
# do the magic
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%
{SERVER_NAME}}/docs/$1

## and now deal with CGIs - we have to force a handler
```

```
RewriteCond %{REQUEST_URI} ^/cgi-bin/  
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/cgi-  
bin/$1 [H=cgi-script]
```



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.map

# deal with aliases as above
RewriteCond %{REQUEST_URI} !^/icons/
RewriteCond %{REQUEST_URI} !^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
# this does the file-based remap
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/docs/$1

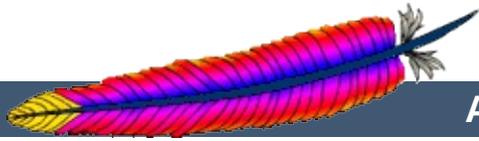
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/cgi-bin/$1 [H=cgi-script]
```

---

Copyright 2017 The Apache Software Foundation.

Licensed under the [Apache License, Version 2.0](#).

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



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)
- [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://product.example.com/widget/$1 [P]
ProxyPassReverse /products/widget/
http://product.example.com/widget/
```

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} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^/(.*) http://old.example.com/$1 [P]
ProxyPassReverse / http://old.example.com/
```

## Discussion:

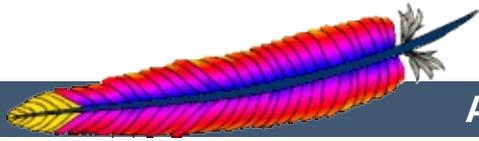
In each case, we add a `ProxyPassReverse` directive to ensure that any redirects issued by the backend are correctly passed on to the client.

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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)
- [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 might define a `RewriteMap` as:

```
RewriteMap examplemap txt:/path/to/file/map.txt
```

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|/not_found.html}
```

### 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.



## Using Plain-Text Maps

When a MapType of `txt` is used, the MapSource is a filesystem path to a plain-text mapping file, containing space-separated key/value pair per line. Optionally, a line may contain a comment, starting with a '#' character.

For example, the following might be valid entries in a map file.

```
# 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 might 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/productmap.txt
RewriteRule ^/product/(.*) /prods.php?
id=${product2id:$1|NOTFOUND} [PT]
```

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|NOTFOUND} [PT]
```

### **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 might use the following map file and directives to provide a random load balancing between several back-end server, 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://${servers:static}/$1  
[NC,P,L]  
RewriteRule ^/(.*) http://${servers:dynamic}/$1 [P,L]
```

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/mapfile.dbm
```

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.map
```

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.



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 (.*[A-Z]+.*) ${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.



## 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.

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 underscores
    print $_;
}
```

### Use a RewriteLock!

When using a prg: RewriteMap, you should use a [RewriteLock](#). Failure to do so will result in an error message

in the log file, and may result in a race condition on concurrent requests.

### **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.



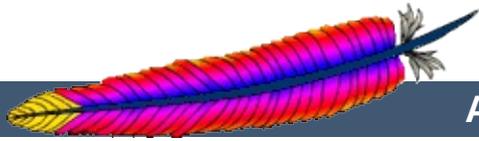
The `RewriteMap` directive can occur more than once. For each mapping-function use one `RewriteMap` directive to declare its rewriting mapfile.

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](#)



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Rewrite](#)

# Advanced Techniques with mod\_rewrite

This document supplements the [mod\\_rewrite reference documentation](#). It provides a few advanced techniques and tricks 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:/path/to/map.users-to-hosts

RewriteRule ^/u/([^/]+)/?(.*) http://${users-to-hosts:$1|server0}/u/$1/$2
```

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$      /regen
```

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.txt

RewriteRule ^/(.*) http://${lb:servers}/$1 [P,L]
```

`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:

Wouldn't it be nice, while creating a complex web page, if the web browser would automatically refresh the page every time we save a new version from within our editor? Impossible?

## Solution:

No! We just combine the MIME multipart feature, the web server NPH feature, and the URL manipulation power of [mod\\_rewrite](#). First, we establish a new URL feature: Adding just `:refresh` to any URL causes the 'page' to be refreshed every time it is updated on the filesystem.

```
RewriteRule ^(/[uqe]/[^\s]+/?.*):refresh
/internal/cgi/apache/nph-refresh?f=$1
```

Now when we reference the URL

```
/u/foo/bar/page.html:refresh
```

this leads to the internal invocation of the URL

```
/internal/cgi/apache/nph-refresh?f=/u/foo/bar/page.html
```

The only missing part is the NPH-CGI script. Although one would usually say "left as an exercise to the reader" ;-) I will provide this, too.

```
#!/sw/bin/perl
##
##  nph-refresh -- NPH/CGI script for auto refreshing pages
##  Copyright (c) 1997 Ralf S. Engelschall, All Rights Reser
##
$| = 1;
```

```

# split the QUERY_STRING variable
@pairs = split( /&/, $ENV{'QUERY_STRING'} );
foreach $pair ( @pairs ) {
    ( $name, $value ) = split( /=/, $pair );
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))
    eval "\$$name = \"$value\"";
}
$QS_s = 1    if ( $QS_s eq '' );
$QS_n = 3600 if ( $QS_n eq '' );
if ( $QS_f eq '' ) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
print "<b>ERROR</b>; No file given\n";
    exit(0);
}
if ( !-f $QS_f ) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
print "<b>ERROR</b>; File $QS_f not found\n";
    exit(0);
}

sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
    print "Content-type: multipart/x-mixed-replace;boundary="
    &print_http_headers_multipart_next;
}

sub print_http_headers_multipart_next {
    print "\n--$bound\n";
}

```

```

}

sub print_http_headers_multipart_end {
    print "\n--$bound--\n";
}

sub displayhtml {
    local ($buffer) = @_;
    $len = length($buffer);
    print "Content-type: text/html\n";
    print "Content-length: $len\n\n";
    print $buffer;
}

sub readfile {
    local ($file) = @_;
    local ( *FP, $size, $buffer, $bytes );
    ( $x, $x, $x, $x, $x, $x, $x, $size ) = stat($file);
    $size = sprintf( "%d", $size );
    open(FP, "&lt;$file");
    $bytes = sysread( FP, $buffer, $size );
    close(FP);
    return $buffer;
}

$buffer = &readfile($QS_f);
&print_http_headers_multipart_begin;
&displayhtml($buffer);

sub mystat {
    local ($file) = $_[0];
    local ($time);

    ( $x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime ) = stat($f

```

```

    return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ( $n = 0 ; $n < lt ; $QS_n ; $n++ ) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ( $mtime ne $mtimeL ) {
            $mtimeL = $mtime;
            sleep(2);
            $buffer = &readfile($QS_f);
            &print_http_headers_multipart_next;
            &displayhtml($buffer);
            sleep(5);
            $mtimeL = &mystat($QS_f);
            last;
        }
        sleep($QS_s);
    }
}

&print_http_headers_multipart_end;

exit(0);

##EOF##

```



## 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]+)(.*)
/home/$2/$1/public_html$3
```



**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} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^foo\.html$ foo.day.html [L]
RewriteRule ^foo\.html$ foo.night.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.

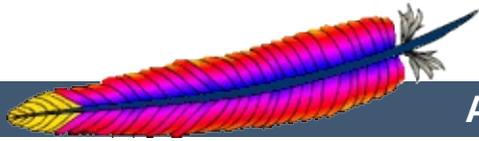
## Solution:

Use the [E] flag to set an environment variable.

```
RewriteEngine on  
RewriteRule ^/horse/(.*) /pony/$1 [E=rewritten:1]
```

Later in your ruleset you might check for this environment variable using a RewriteCond:

```
RewriteCond %{ENV:rewritten} =1
```



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)



`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 `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.



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.

Third-party 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.



**Complete Example**

RewriteRule provides the `[P]` flag to pass rewritten URIs through `mod_proxy`.

```
RewriteRule ^/?images(.*) http://imageserver.local/images$1 [P]
```

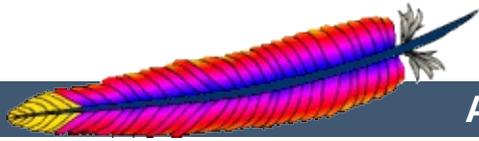
However, in many cases, when there is no actual pattern matching needed, as in the example shown above, the `ProxyPass` directive is a better choice. The example here could be rendered as:

```
ProxyPass /images/ http://imageserver.local/images/
```

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://imageserver.local/images/
```

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.



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



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [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 and tricks](#)

[When not to use mod\\_rewrite](#)



The internal processing of this module is very complex but needs to be explained once even to the average user to avoid common mistakes and to let you exploit its full functionality.



First you have to understand that when Apache processes a HTTP request it does this in phases. A hook for each of these phases is provided by the Apache API. Mod\_rewrite uses two of these hooks: the URL-to-filename translation hook which is used after the HTTP request has been read but before any authorization starts and the Fixup hook which is triggered after the authorization phases and after the per-directory config files (`.htaccess`) have been read, but before the content handler is activated.

So, after a request comes in and Apache has determined the corresponding server (or virtual server) the rewriting engine starts processing of all mod\_rewrite directives from the per-server configuration in the URL-to-filename phase. A few steps later when the final data directories are found, the per-directory configuration directives of mod\_rewrite are triggered in the Fixup phase. In both situations mod\_rewrite rewrites URLs either to new URLs or to filenames, although there is no obvious distinction between them. This is a usage of the API which was not intended to be this way when the API was designed, but as of Apache 1.x this is the only way mod\_rewrite can operate. To make this point more clear remember the following two points:

1. Although mod\_rewrite rewrites URLs to URLs, URLs to filenames and even filenames to filenames, the API currently provides only a URL-to-filename hook. In Apache 2.0 the two missing hooks will be added to make the processing more clear. But this point has no drawbacks for the user, it is just a fact which should be remembered: Apache does more in the URL-to-filename hook than the API intends for it.
2. Unbelievably mod\_rewrite provides URL manipulations in per-directory context, *i.e.*, within `.htaccess` files, although these are reached a very long time after the URLs have been translated to filenames. It has to be this way because

.htaccess files live in the filesystem, so processing has already reached this stage. In other words: According to the API phases at this time it is too late for any URL manipulations. To overcome this chicken and egg problem mod\_rewrite uses a trick: When you manipulate a URL/filename in per-directory context mod\_rewrite first rewrites the filename back to its corresponding URL (which is usually impossible, but see the RewriteBase directive below for the trick to achieve this) and then initiates a new internal sub-request with the new URL. This restarts processing of the API phases.

Again mod\_rewrite tries hard to make this complicated step totally transparent to the user, but you should remember here: While URL manipulations in per-server context are really fast and efficient, per-directory rewrites are slow and inefficient due to this chicken and egg problem. But on the other hand this is the only way mod\_rewrite can provide (locally restricted) URL manipulations to the average user.

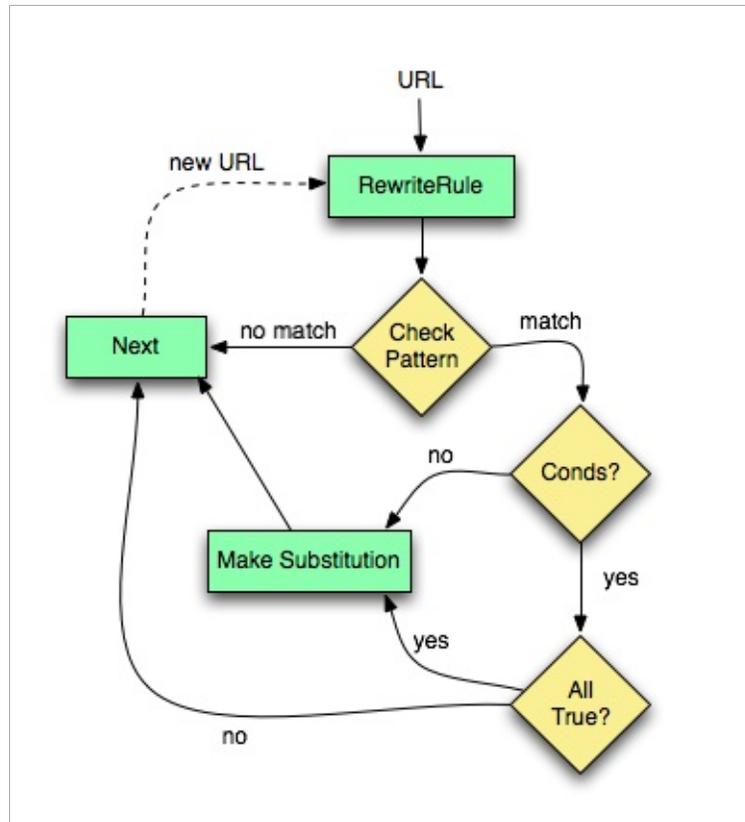
Don't forget these two points!



## 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

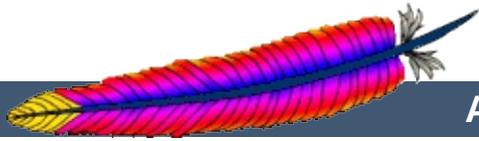
*control flow through the rewriting ruleset*

As you can see, first the URL is matched against the *Pattern* of each rule. When 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. If none are present, it just 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](#)



## Apache HTTP Server Version 2.2

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.2](#) > [Developer Documentation](#)

## Apache 2.0 Thread Safety Issues

When using any of the threaded mpms in Apache 2.0 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 conditons 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/mysqlclient/">http://dev.mysql.com/doc/mysqlclient/</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_CRYPTO_set_locking_callback CRYPTO_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 threads crypt() calls
<a href="#">Sablotron</a>	0.95	?	
<a href="#">zlib</a>	1.1.4	Yes	Relies upon thread-safe zalloc and 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](#)