

| | [FAQ](#) | |



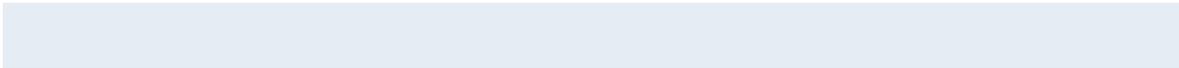
Apache HTTP Server Version 2.0

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

Apache HTTP Server Version 2.0

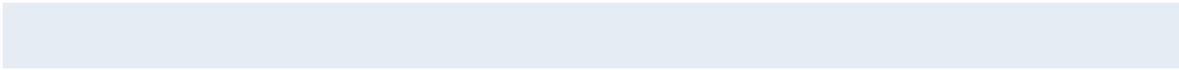


Google



2.0

1.3 2.0



(MPM)



(content negotiation)

(DSO)

URL

SSL/TLS

CGI Suexec

URL (rewriting)

How-To /

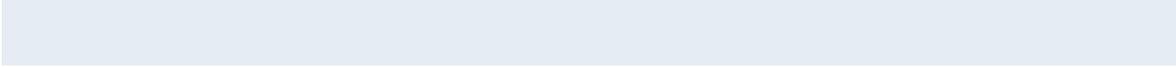
[...](#)

[CGI:](#)

[.htaccess](#)

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

[\(public_html\)](#)

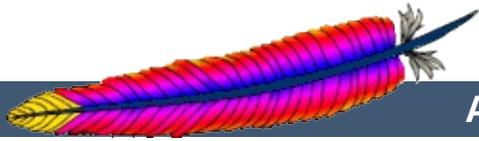


[Microsoft Windows](#)

[Novell NetWare](#)

[EBCDIC](#)

(FAQ)



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

1.3 2.0

src/CHANGES

2.0



-
- `autoconf libtool` .
Apache 1.3 APACI .
 - Apache 2.0
[Processing Modules](#) (MPM) .



- Apache 1.3 `MPM . Apache 1.3`
`prefork MPM . MPM`

- `proxy` HTTP/1.1 .
`<Directory proxy:> <Proxy> .`

- `PATH_INFO () .`
`PATH_INFO .`
`PHP core PATH_INFO . core`
`PATH_INFO server-side include PATH_`
`, AcceptPathInfo .`

- `CacheNegotiatedDocs` on off .
`CacheNegotiatedDocs CacheNegotiatedDocs on`

- `ErrorDocument`

```
ErrorDocument 403 "Some Message"
```

```
ErrorDocument 403 "Some Message"
```

URL

- `AccessConfig ResourceConfig .`
`Include .`
`http.conf Include conf/access.conf Include`
`conf/srm.conf .`
`Include httpd.conf , srm.conf access.co`
- `BindAddress Port . Listen .`
- Apache-1.3 Port URL . Apache-
2.0 `ServerName .` URL

- ServerType . MPM . inetd
MPM .
- AgentLog, RefererLog, RefererIgnore
mod_log_agent mod_log_referer .agent
referer [mod_log_config](#) [CustomLog](#) .
- AddModule ClearModuleList .
. Apache 2.0 API ,
- FancyIndexing . [IndexOptions](#)
FancyIndexing .
- [mod_negotiation](#) MultiViews .
. [MultiviewsMatch](#) .
- (2.0.51)
ErrorHandler , [Hea](#)

Header always set



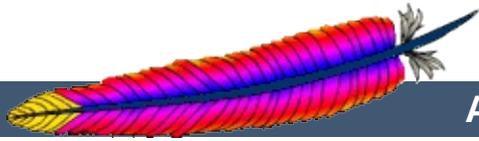
-
- Apache 1.3 `mod_auth_digest` .
 - Apache 1.3 `mod_mmap_static` `mod_file_cache`
 - `src` .



Apache 2.0 API . Apache 1.3 API
Apache 2.0 . [Apache 2.0](#) .

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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

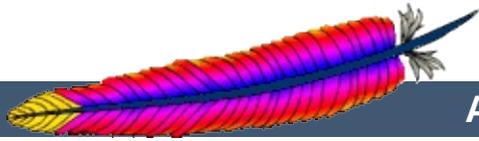
HTML ,

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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.

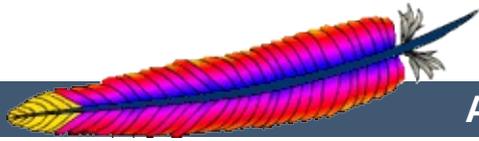
```
Copyright [yyyy] [name of copyright owner]
```

```
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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



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_0_22.tar.gz
$ tar xvf httpd-2_0_22.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 .

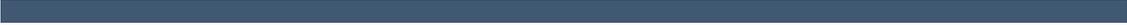
(5.003 .) ` configure' 2.0

. Perl (

Perl 4 Perl 5) ./configure --

perl () .





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

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



tar :

```
$ gzip -d httpd-2_0_MN.tar.gz  
$ tar xvf httpd-2_0_MN.tar
```



```

configure .(          CVS
libtool,          buildconf
.)

./configure .

./configure .

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

configure ,
configure .      configure manp

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

```

```

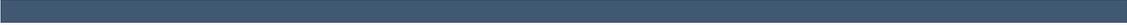
$ CC="pgcc" CFLAGS="-O2" \
./configure --prefix=/sw/pkg/apache \
--enable-rewrite=shared \
--enable-speling=shared

```

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/2.0/> [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 , , :

```

```

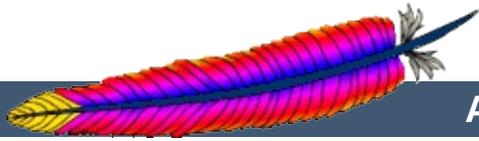
$ ./config.nice
$ make
$ make install
$ PREFIX/bin/apachectl stop
$ PREFIX/bin/apachectl start

```

```

prefix( Listen )
.

```



| | [FAQ](#) | |



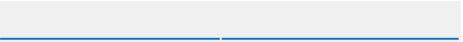
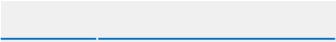
Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



Windows NT, 2000, XP ,

Windows 95 ME .



[httpd](#)

h



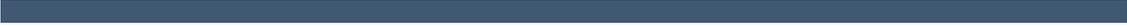
[httpd](#)

[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          ()
```



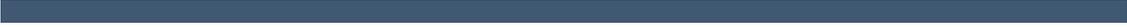


```
'  
    "    Unable to bind to Port ...".  
:
```

- root .
- .

[FAQ](#) .





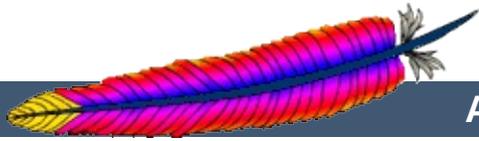
```
, ( rc.  
apachectl . root .  
.  
apachectl SysV init .  
restart, stop httpd . a  
init . .
```



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

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



. 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 .
apachectl .
```

```
httpd , :
```

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

```
ServerRoot PidFile .
```



: TERM

apachectl -k stop

TERM stop

.

.

,

.



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

```
USR1 graceful
    )
```

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

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

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

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

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

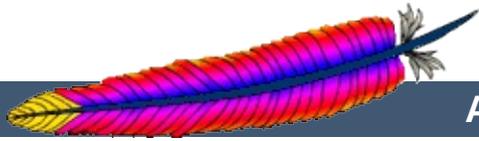
scoreboard . [ScoreBoa](#)

HTTP (KeepAlive)

. 1.2

KeepAlive

20



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

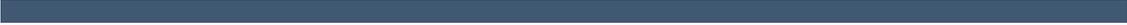
[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

--



.





```
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>
```

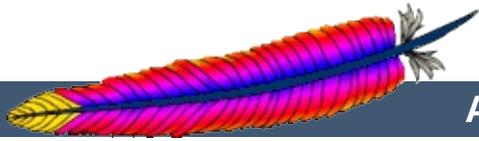
<DirectoryMatch>, <Files>, <FilesMatch>, <Location>,
<LocationMatch> .

Directory, Location, Files .



```
AccessFileName  
AllowOverride
```

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



, , , , URL .

.htaccess .



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

```

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

```

```

<IfDefine> httpd .
,          httpd -DClosedForNow
:

```

```

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

```

```

<IfModule>

```

```

.
.
.

```

```

mod_mime_magic MimeMagicFiles .

```

```

<IfModule mod_mime_magic.c>

```

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

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

.



```

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

```

```

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

```

```

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

```

```

<Files> .
,          private.html .

```

```

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

```

```

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

```

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

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

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

<Location> . URL
mod_status . server-status
 .

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

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

<LocationMatch> .

.
:

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

:

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

<Directory> <Files> . ()

<Location> .

<Location> . (URL)

, . :

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

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

?

http://yoursit

.

<Directory>

.(. .

<Directory> .

Options

.)

.
.
.

<Location /> URL



<VirtualHost> .



<Proxy> <ProxyMatch> URL

mod_proxy

cnn.cc

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



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

.

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



```

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

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

```

```

    <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>
., !

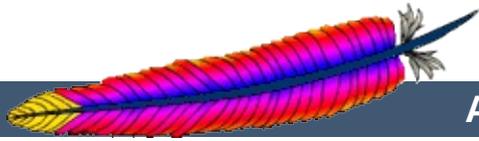
```

```

<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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

core .



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

ServerAdmin ServerTokens

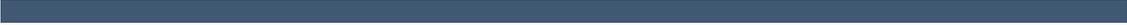
. ServerTokens HTTP .

ServerName UseCanonicalName URL .

,

.





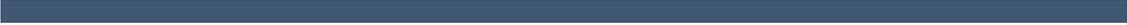
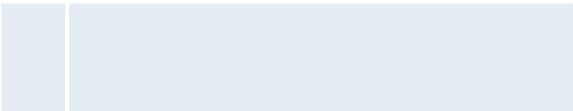
- CoreDumpDirectory
- DocumentRoot
- ErrorLog
- LockFile
- PidFile
- ScoreBoardFile
- ServerRoot

. root

. (/)

.



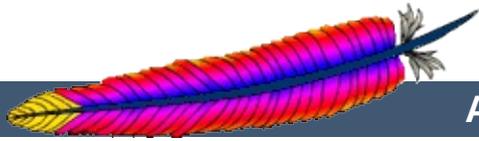



LimitRequestBody
LimitRequestFields
LimitRequestFieldsize
LimitRequestLine
RLimitCPU
RLimitMEM
RLimitNPROC
ThreadStackSize

LimitRequest* .
of service) .

RLimit* . CGI
.

ThreadStackSize Netware .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

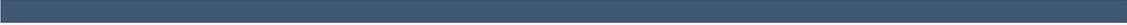
[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



.

.





(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](#)



mod_cgi	PidFile
mod_rewrite	RewriteLog
	RewriteLogLevel
	ScriptLog
	ScriptLogBuffer
	ScriptLogLength

PID

logs/httpd.pid httpd process id .

[PidFile](#) .process-id

. -k .

[ScriptLog](#)

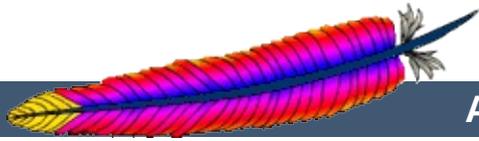
CGI .

. . [mod_cgi](#) .

[mod_rewrite](#)

[Rewri](#)

. [RewriteLogLevel](#) .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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 speling</u>	<u>ErrorDocument</u>
<u>mod vhost alias</u>	<u>Options</u>
	<u>ProxyPass</u>
	<u>ProxyPassReverse</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

. . internal.exam
internal.example.com



(Rewriting Engine)

mod_rewrite .

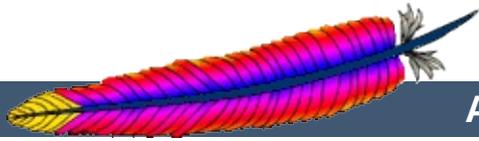
., mod_rewrite

(alias), , , . mod_rewrite



File Not Found

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

--



.

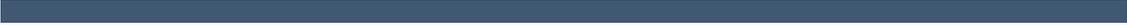
.

.

,

.





. , 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)

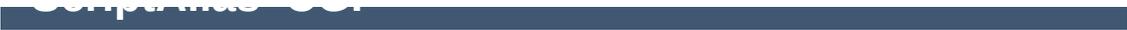
```
. SSI SSI
,
, SSI CGI . SSI "exec cmd"
httpd.conf CGI
SSI
SSI CGI
.html .htm SSI .
. SSI .shtml
SSI
IncludesNOEXEC . ScriptAlias Q
#include virtual="..." --> CGI . <
```



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](#)

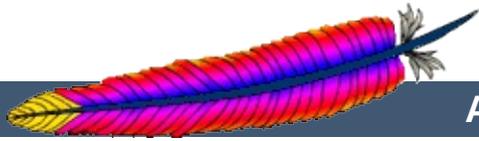
10 :

```
[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>
```



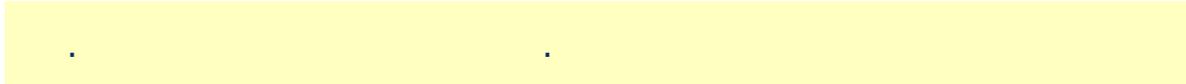
| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

(DSO)



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

.
DSO .



```
mod_so LoadModule
```

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



Apache 2.0 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. :

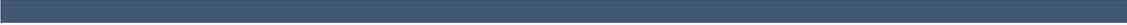
```
$ ./configure --enable-so  
$ make install
```

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

.

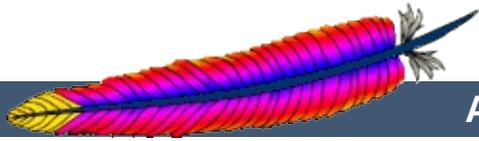


DSO :

- `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 DSO (ld -lfoo) () DSO`
`ELF a.out) DSO`
`DSO C (`
`/, (li`
`, dlopen() .`



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

(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` , .
`(JPEG, GIF, ASCII-art) media type`
`(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=en" prefer-language=en
SetEnvIf Cookie "language=fr" prefer-language=fr

```



(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)

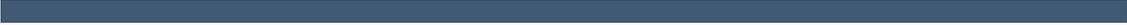
,

html

.

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





URL .

URL .

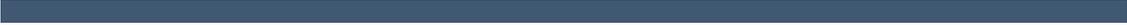
HTTP/1.1

HTTP/1.0 .,

CacheNegotiatedDocs HTTP/1.0 ()

HTTP/1.1 .





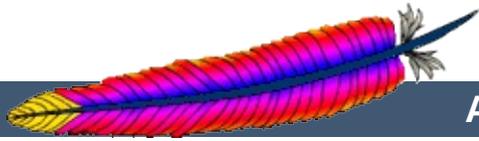
Alan J. Flavell

[Language Negotiation Notes](#)

2.0

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



"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 . .

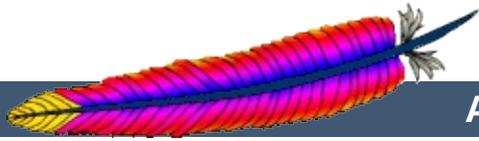
CGI REDIRECT_ . REDIRECT_
REDIRECT_HTTP_USER_AGENT . , HTTP_USER_AGE
REDIRECT_URL
REDIRECT_URL REDIRECT_STATUS . URL URL

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

(Binding)

.

.

DNS



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

IP,

Listen

. Listen

List

, 80 8000

:

```
Listen 80
Listen 8000
```

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

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 .
```

```
IPv4 IPv6 IPv4- IPv6 .
--enable-v4-mapped , Listen
```

:

```
Listen 80
```

```
--enable-v4-mapped Listen .
enable-v4-mapped FreeBSD, NetBSD, OpenBSD
```

,

```
APR IPv4 , Listen IPv4
```

:

```
Listen 0.0.0.0:80
Listen 192.170.2.1:80
```

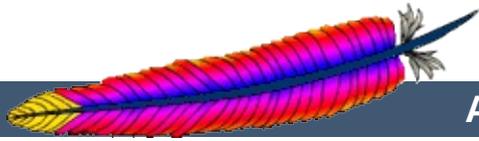
```
IPv4 IPv6 , --dis
mapped Listen :
```

```
Listen [::]:80
Listen 0.0.0.0:80
```

```
--disable-v4-mapped Listen .
--disable-v4-mapped FreeBSD, NetBSD, OpenBSD .
```



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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

(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 .

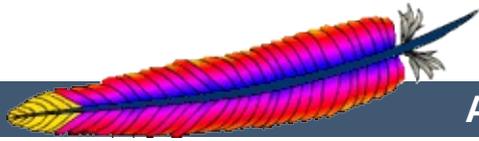
./httpd -l MPM . MPM

.



MPM . MPM .

BeOS	<u>beos</u>
Netware	<u>mpm_netware</u>
OS/2	<u>mpmt_os2</u>
	<u>prefork</u>
	<u>mpm_winnt</u>



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



(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_access</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 ,
 .

Header

HTTP

mod_ext_filter ExtFilterDefine
disableenv= enableenv=

URL (Rewriting)

RewriteCond *TestString* %{ENV:...} mod_rewrite
mod_rewrite ENV:
mod_rewrite



`SetEnv PassEnv`

downgrade-1.0

HTTP/1.0

force-no-vary

Vary

`ForceNoVary`

`force-respo`

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

no-gzip

`mod_deflate` DEFLATE

nokeepalive

`KeepAlive`

prefer-language

`mod_negotiation` (

en, ja, x-klinc

`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
```

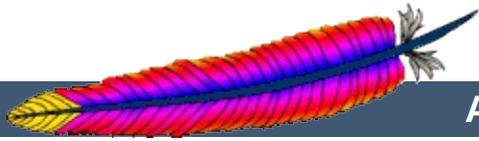
..

/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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



.



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](#) .

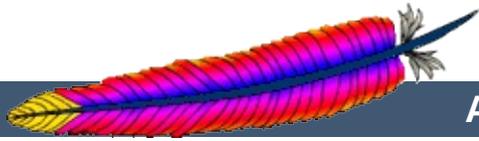
request

```
char *handler
```

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

```
. content type .
```

```
type .
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



.



<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 .

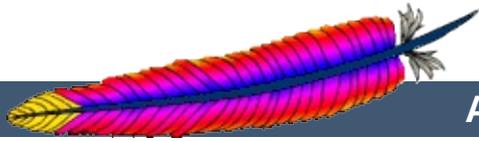
INCLUDES

mod include Server-Side Includes

DEFLATE

mod deflate

, mod ext filter .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

suEXEC

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



setuid setgid
suEXEC

, setui

, 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



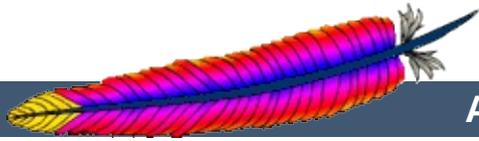
! . suEXEC ""
wrapper .

- suEXEC
-

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

- suEXEC PATH

- suEXEC



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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_SCOREBOARD
USE_SHMGET_SCOREBOARD . ( HAVE_SHMGET
HAVE_SHMGET )
src/main/http_main.c (hook) .
( .)

```

```

: 1.2
.

```

DYNAMIC_MODULE_LIMIT

```

(
-DDYNAMIC_MODULE_LIMIT=0 .

```



Solaris 8 worker MPM 2.0.38

(trace).

:

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

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

strace, ktrace, par . .

10KB .

().

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

(listener) LWP #67 .

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

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

(accept) worker .
worker LWP #65 .

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

(local) . (

)

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

brk(2) (heap) .
apr_bucket_alloc)

malloc(3) .

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

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

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

worker .

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

Options FollowSymLinks AllowOverride None.

lstat(2) .htaccess .

1) , 2) , stat(2) .

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

sendfilev(2) HTTP .

Sendfile . sendfile(2)

write(2) writev(2) .

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

```

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

```

```

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

```

worker (lingering close).

```

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

```

worker , (listener)

```

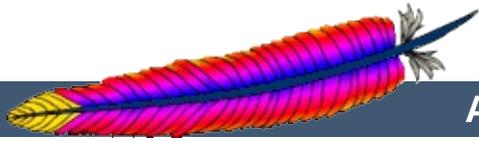
/67: accept(3, 0x001FEB74, 0x001FEB94, 1) (sleeping...)

```

```

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

```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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/$2 [R]  
RewriteRule ^/([uqe])/([^/]+)$ /$1/$2/ [R]
```

```
:  
    . ,  
example.com www.example.com
```

```
.
```

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

DocumentRoot

```
:  
    DocumentRoot URL "/" .  
    , ( Docu  
    /e/www/ ( ) /e/www/ .  
    /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 .  
    , CGI URL 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_imap

:

```
NCSA . NCSA
      mod_imap .
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 .

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

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

```

```

:
    ourdomain.com          URL
www.somewhere.com        . .

```

```

:
:

```

```

RewriteEngine on
RewriteCond    %{REMOTE_HOST}    !^\.+\.ourdomain\.com$
RewriteRule    ^(/~.+)*          http://www.somewhere.com/$1 [R

```

URL

```

:
URL A          B
Perl          ErrorDocument CGI          ,          mod_re
.             ErrorDocument CGI          !

```

```

:
:

```

```

RewriteEngine on
RewriteCond    /your/docroot/%{REQUEST_FILENAME}    !-f
RewriteRule    ^(.+)                                http://webse

```

DocumentRoot . (

) , :

```
RewriteEngine on
RewriteCond %{REQUEST_URI} !-U
RewriteRule ^(.+) http://webserverB.dom/$1
```

mod_rewrite URL (look-ahead) . URL
. CPU
. ErrorDocument C

:
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 redirect
## Copyright (c) 1997 Ralf S. Engelschall, All Rights Reserved
##
```

```

$| = 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

```

```

URL foo.html      07:00-19:00    foo.day.html
,      foo.night.html .
...

```

YYYY XXXX

```

:
    .html .phtml      document.YYYY
document.XXXX (backward compatibility) URL
)      ?

```

```

:
    . ,
.

```

```

# .html
# .phtml
# .html .phtml
#
RewriteEngine on
RewriteBase    /~quux/
# ,
RewriteRule    ^(.*)\.html$           $1      [C,E=wasHTML
# .phtml
RewriteCond    %{REQUEST_FILENAME}.phtml -f
RewriteRule    ^(.*)$ $1.phtml        [S=1]
#
RewriteCond    %{ENV:wasHTML}         ^yes$
RewriteRule    ^(.*)$ $1.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.dom) .
```

:

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

mod_rewrite : proxy throughput

```
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
```

```
:
URL CGI , MIME-type          CGI .
/~quux/foo.html              /~quux/foo.cgi .
```

```
RewriteEngine on
RewriteBase    /~quux/
RewriteRule    ^foo\.html$ foo.cgi [T=application/x-httpd-
```

```
:
: ,                               ,
) CGI .                            ( cron
```

```
:
:
```

```
RewriteCond %{REQUEST_FILENAME}  !-s
RewriteRule ^page\.html$         page.cgi [T=application/
```

```
page.html          page.html 0
.                  page.cgi CGI          STDO
page.html .        page.html .
, (cron)           page.html .
```

```
:
```

```
:
```

```
! MIME multipart NPH ,          mod_rewrite U
. , URL : URL                    :refresh
```

```
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_URI} !^/commonurl1/. *
RewriteCond %{REQUEST_URI} !^/commonurl2/. *
#
RewriteCond %{REQUEST_URI} !^/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
:

```



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

```
:
(                                     ) /~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 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 .



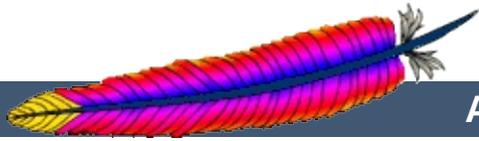
:
FAQ: ? [mod_rewrite](#) ...

:
[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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

IP

ServerPath



IP IP

HTTP .

. IP

.

DNS

IP ,

.

IP .

. IP :

•

HTTP/1.1 , HTTP/1.0

• SSL SSL

• IP

(bandwidth)



```

core DocumentRoot
      NameVirtualHost
      ServerAlias
      ServerName
      ServerPath
      <VirtualHost>

```

```

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

```

```

      ServerName DocumentRoot ServerName <v
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 .

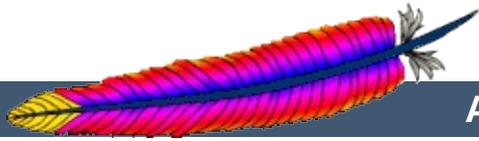
ServerPath :

:

```
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](#) | |

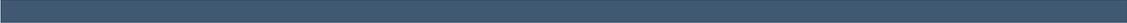


Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

IP





.

.

:

- 2 1

User, Group, Listen, ServerRoot

- , IP

Listen .

, (
)

(file descriptor)

:

- .
- .



. [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>

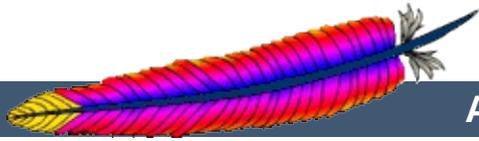
<VirtualHost www.baygroup.org>
ServerAdmin webmaster@mail.baygroup.org
DocumentRoot /groups/baygroup/www
ServerName www.baygroup.org
ErrorLog /groups/baygroup/logs/error_log
TransferLog /groups/baygroup/logs/access_log
</VirtualHost>
```

IP . ([DNS](#))

VirtualHost
VirtualHost

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

:
.



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >



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 :
```

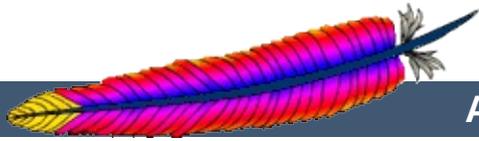
```
RewriteEngine on

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

[Empty rectangular box]



IP



IP , DNS (CNAMEs)
www.example1.com www.example2.org .

Note

DNS
hosts
hosts

```
# 80
Listen 80

# IP
NameVirtualHost *:80

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

<VirtualHost *:80>
  DocumentRoot /www/example2
  ServerName www.example2.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.example1.com

  # ...
</VirtualHost>

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

  # ...
</VirtualHost>
```

172.20.30.50 . ,
172.20.30.50 www.example1.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.example1.com
  DocumentRoot /www/domain-80
</VirtualHost>

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

<VirtualHost 172.20.30.40:80>
  ServerName www.example2.org
  DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  ServerName www.example2.org
  DocumentRoot /www/otherdomain-8080
</VirtualHost>
```



```
www.example1.com www.example2.org IP  
(172.20.30.40 172.20.30.50).
```

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

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

```
.
```



www.example1.com www.example2.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.example1.com
</VirtualHost>

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

<VirtualHost 172.20.30.50:80>
  DocumentRoot /www/example2-80
  ServerName www.example1.org
</VirtualHost>

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



, IP

```
Listen 80

NameVirtualHost 172.20.30.40

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

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

sub1-

Host: ,

URL http://www.sub1.dc

sub1-

Host:

:

Host:

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

sub1-

RewriteRule

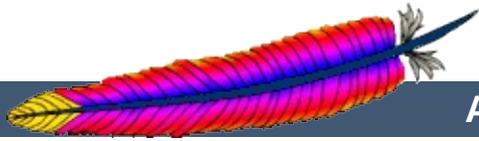
Host:

(

URL .

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >



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: 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

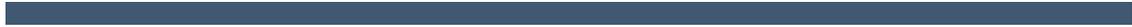
VirtualHost

.

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_

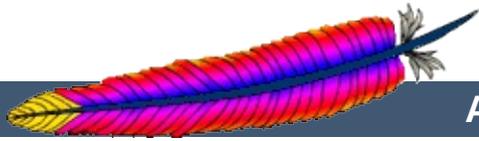
- DNS VirtualHost DNS .
DNS .
- ServerName . DNS .



DNS

:

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

(file descriptor)

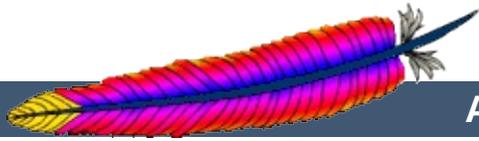
handle)) . , (file
10-20 . ,
hard-limit .
:
1. setrlimit() .
2. (Solaris 2.3) setrlimit(RLIMIT_NOFILE)
3. hard limit .
4. (Solaris 2) stdio 256

:

- . <VirtualHost> . (
-) 12 ,

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





| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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 10.0.0.1 . :

```
<VirtualHost 10.0.0.1>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```

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

```
<VirtualHost 10.0.0.1>
ServerName www.abc.dom
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>
```



```

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

```

```

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

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

```

```

www.abc.dom 10.0.0.1, www.def.dom 10.0.0.2 .
, def.dom DNS . def.dom
. www.def.dom 10
. DNS www.de
.

```

```

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

```



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

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

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



-
- VirtualHost IP

- Listen IP

- ServerName

- `<VirtualHost _default_*>`



DNS . 1.2
. IP

DNS

IP DNS

DNS . (FTP
)

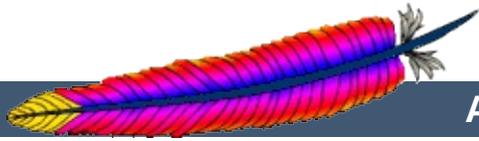
TCP wrapper "-" DNS

IP DNS

HTTP/1.1

Host IP

DNS . 1997 3



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [FAQ](#)

- (FAQ)



- [" ... ? ... ?"](#)
- [?](#)

" ... ? ... ?"

:

(errorlog) !

(
 /usr/local/apache2/logs/error_log,
[ErrorLog](#) .

[FAQ](#) !

FAQ

(The Apache Group)
 (open) (closed)

"

USENET :

- [comp.infosystems.www.servers.unix](#) [[news](#)] [[google](#)]
- [comp.infosystems.www.servers.ms-windows](#) [[news](#)] [[google](#)]
- [comp.infosystems.www.authoring.cgi](#) [[news](#)] [[google](#)]

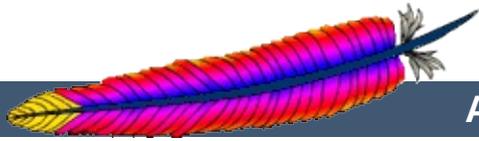
```
core dump () backtrace(;
```

```
# cd ServerRoot  
# dbx httpd core  
(dbx) where
```

```
(ServerRoot, httpd, core . dbx gd  
.)
```

?

40



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [FAQ](#)

- (FAQ)



- [Invalid argument: core_output_filter: writing data to the network](#)
- [AcceptEx failed](#)
- [Premature end of script headers](#)

Invalid argument: core_output_filter: writing data to the network

```

sendfile ,
sendfile
(error log) 0 0
sendfile
sendfile EnableSendfile .,
EnableMMAP .

```

AcceptEx Failed

```

win32 AcceptEx , Win32DisableAcc

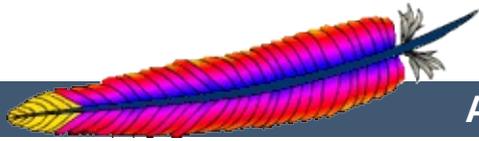
```

Premature end of script headers

```

CGI Internal Server Error
CGI

```



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 permission by the author, from the article [Introducing SSL and Certificates using SSLeay](#) from [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 except by those it is intended for. Once in this form, the message may only be interpreted through the use of 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. If this key is secret, then nobody other than the sender or receiver may read the message. If Alice and the bank know a secret key, then they may send each other private messages. The task of privately choosing a key before communicating, however, 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 may 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 may send private messages to the owner of a key-pair (the bank), by encrypting it using their public key. Only the bank will be able to decrypt it.

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 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 they agree then the message was received intact.

A summary such as this is called a *message digest*, *one-way function* or *hash function*. Message digests are used to create short, fixed-length representations of longer, variable-length messages. Digest algorithms are designed to produce unique digests for different messages. Message digests are designed to make it too difficult to determine the message from the digest, and also 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; when this is achieved, the integrity of the associated message is assured. One way to do this is to include the digest 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 does not 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 may *decrypt* the signature using the public key, only the signer knows the private key. This means that only they may have signed it. 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 corresponds to the bank's private key. Similarly, the bank also needs to verify that the message signature really corresponds to Alice's signature.

If each party has a certificate which validates the other's identity, confirms the public key, and is signed by a trusted agency, then they both will 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

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

etc.

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

Table 2: Distinguished Name Information

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

A Certificate Authority may define a policy specifying which distinguished field names are optional, and which are required. It may also place requirements upon the field contents, as may users of certificates. As an example, a Netscape browser requires that the Common Name for a certificate representing a server has a name which 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]. This encoded version is called PEM encoded (the name comes from "Privacy Enhanced Mail"), when placed between begin and end delimiter lines as illustrated in the following example.

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

```
-----BEGIN CERTIFICATE-----
MIIC7jCCAlEgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBqTELMakGA1UEBhMCWFkx
FTATBgNVBAGTDFNuYWt1IERlc2VydDETMBEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChM0U25ha2UgT21sLCBMdGQxHjAcBgNVBAsTFUNlcnRpZm1jYXR1IEF1dGhv
cm10eTEVMBMGA1UEAxMMU25ha2UgT21sIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTgxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAkGA1UEBhMCWFkxFTATBgNVBAGTDFNuYWt1IERlc2VydDETMBEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChM0U25ha2UgT21sLCBMdGQxHjAcBgNVBAsTD1d1YnNl
cnZlc1BUZWFtMRkwFwYDVQDEExB3d3cuc25ha2VvaWwuZG9tMR8wHqYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zcH+da+rPTx/DPRp3xGjHZ4GG6pCmvADIEtBtKBFACZ64n+Dy7Np8b
vKR+yy5DGQiijsH1D/j8H1GE+q4TZ80Fk7BNBFazHxFbYI40KMiCxdKzdif1yfaa
lWoANFlAz1SdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWCGSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAA0B
gQAZUIHAL4D09oE6Lv2k56Gp380BDuILvLg1v1KL8mQR+KFjghCrtpqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdkPfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHZICxBVC11nHyYGjDuAMhe3961YAn8bCl1d/L4NMGBCQ==
-----END CERTIFICATE-----
```

Certificate Authorities

By first verifying the information in a certificate request before granting the certificate, the Certificate Authority assures 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.

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: Since this is who vouches 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. As a result, one must 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. Browsers are preconfigured to trust well-known certificate authorities.

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 how long certificates are valid, they renew them, and they keep lists of certificates that have already been issued but are no longer valid (Certificate Revocation Lists, or CRLs). Say Alice is entitled to a certificate as an employee of a company. Say too, that the certificate needs to be revoked when Alice leaves the company. Since certificates are objects that get passed around, it is impossible to tell from the certificate alone that it has been revoked. When examining certificates for validity, therefore, it is necessary to contact the issuing Certificate Authority to check CRLs -- this is not usually an automated part of the process.

Note

If you use a Certificate Authority that is not configured into browsers 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 at the start of establishing a protocol session.

Table 4: Versions of the SSL protocol

Version	Source	Description	Browser Support
SSL v2.0	Vendor Standard (from Netscape Corp.) [SSL2]	First SSL protocol for which implementations exists	- NS Navigator 1.x/2.x - MS IE 3.x - Lynx/2.8+OpenSSL
SSL v3.0	Expired Internet Draft (from Netscape Corp.) [SSL3]	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	Revision of SSL 3.0 to update the MAC layer	- Lynx/2.8+OpenSSL

	Standard (from IETF) [TLS1]	to HMAC, add block 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).

Session Establishment

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. Though 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, thus avoiding the performance penalty of repeating the many steps needed to start a session. For this the server assigns each SSL session a unique session identifier which is cached in the server and which the client can use on forthcoming

connections to reduce the handshake (until the session identifier expires in the cache of the server).

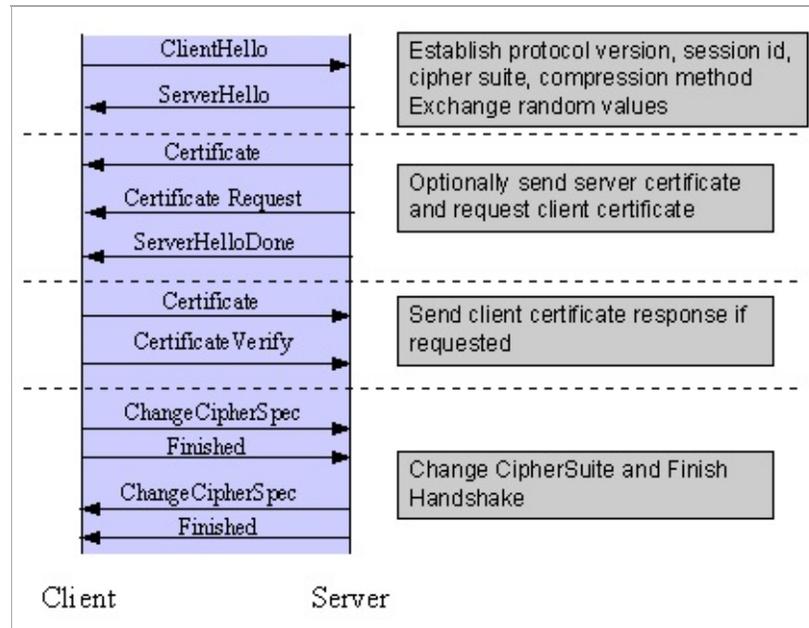


Figure 1: Simplified SSL Handshake Sequence

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 supportable 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 the RSA key exchange when certificates are used, and Diffie-Hellman key exchange for exchanging keys without certificates and 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 assurance against a man-in-the-middle-attack during the information exchange used in generating the shared key [[AC96](#), p516].

Cipher for Data Transfer

SSL uses the conventional cryptography algorithm (symmetric cryptography) described earlier for encrypting messages in a session. There are nine choices, including the choice to perform no encryption:

- 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)

Here "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 one of the best and cryptographically strongest available algorithms, 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 provide integrity and to prevent 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.

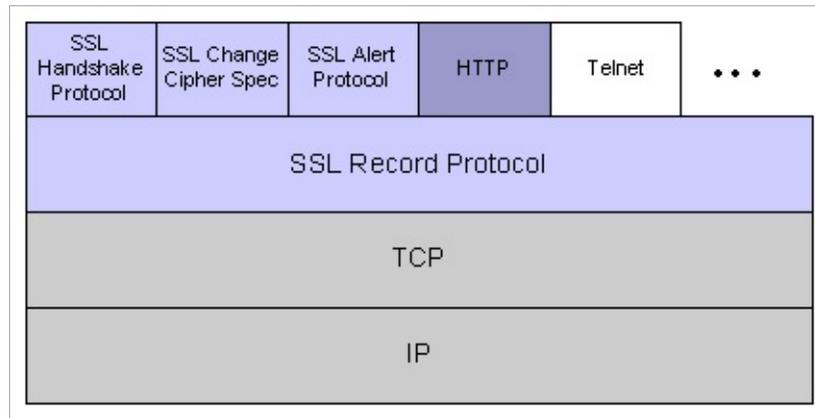


Figure 2: *SSL Protocol Stack*

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 were no session before, then the Null cipher suite is used, which means there is no encryption and messages 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, possibly 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 all major SSL implementations lack support for compression).

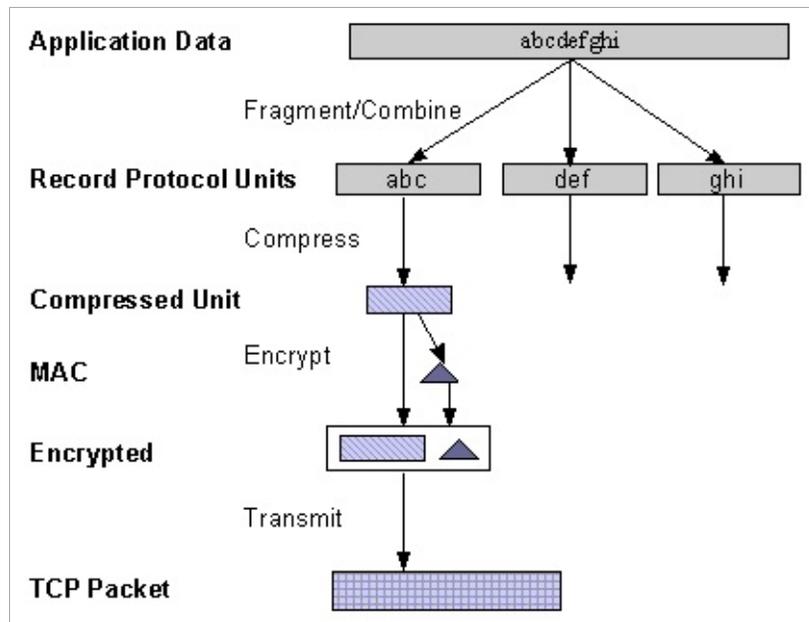


Figure 3: SSL Record Protocol

Securing HTTP Communication

One common use of SSL is to secure Web HTTP communication between a browser and a webserver. This case does not preclude the use of non-secured HTTP. The secure version is mainly plain HTTP over SSL (named HTTPS), but with one major difference: it uses the URL scheme `https` rather than `http` and a different server port (by default 443). This mainly is what [mod_ssl](#) provides to you 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.

[SSL3]

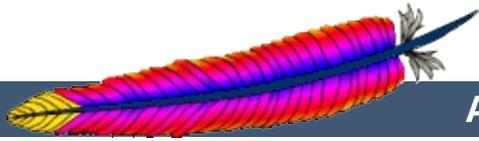
Alan O. Freier, Philip Karlton, Paul C. Kocher, *“The SSL Protocol Version 3.0”*, 1996. See <http://www.netscape.com/eng/ssl3/draft302.txt>.

[TLS1]

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

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [SSL/TLS](#)

SSL/TLS Strong Encryption: Compatibility

All PCs are compatible. But some of them are more compatible than others.

-- Unknown

Here we talk about backward compatibility to other SSL solutions. As you perhaps know, mod_ssl is not the only existing SSL solution for Apache. Actually there are four additional major products available on the market: Ben Laurie's freely available [Apache-SSL](#) (from where mod_ssl were originally derived in 1998), Red Hat's commercial [Secure Web Server](#) (which is based on mod_ssl), Covalent's commercial [Raven SSL Module](#) (also based on mod_ssl) and finally C2Net'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).

The idea in mod_ssl is mainly the following: because mod_ssl provides mostly a superset of the functionality of all other solutions we can easily provide backward compatibility for most of the cases. Actually there are three compatibility areas we currently address: configuration directives, environment variables and custom log functions.



Configuration Directives

For backward compatibility to the configuration directives of other SSL solutions we do an on-the-fly mapping: directives which have a direct counterpart in `mod_ssl` are mapped silently while other directives lead to a warning message in the logfiles. The currently implemented directive mapping is listed in [Table 1](#). Currently full backward compatibility is provided only for Apache-SSL 1.x and `mod_ssl` 2.0.x. Compatibility to Sioux 1.x and Stronghold 2.x is only partial because of special functionality in these interfaces which `mod_ssl` (still) doesn't provide.

Table 1: Configuration Directive Mapping

Old Directive	<code>mod_ssl</code> Directive
Apache-SSL 1.x & <code>mod_ssl</code> 2.0.x compatibility:	
<code>SSLEnable</code>	<code>SSL Engine on</code>
<code>SSLDisable</code>	<code>SSL Engine off</code>
<code>SSLLogFile <i>file</i></code>	<code>SSLLog <i>file</i></code>
<code>SSLRequiredCiphers <i>spec</i></code>	<code>SSLCipherSuite <i>spec</i></code>
<code>SSLRequireCipher <i>c1</i> ...</code>	<code>SSLRequire % {SSL_CIPHER} in {"<i>c1</i>", ...}</code>
<code>SSLBanCipher <i>c1</i> ...</code>	<code>SSLRequire not (% {SSL_CIPHER} in {"<i>c1</i>", ...})</code>
<code>SSLFakeBasicAuth</code>	<code>SSLOptions +FakeBasicAuth</code>
<code>SSLCacheServerPath <i>dir</i></code>	-
<code>SSLCacheServerPort <i>integer</i></code>	-
Apache-SSL 1.x compatibility:	
<code>SSLExportClientCertificates</code>	<code>SSLOptions +ExportCertData</code>

SSLCacheServerRunDir <i>dir</i>	-
Sioux 1.x compatibility:	
SSL_CertFile <i>file</i>	SSLCertificateFile <i>file</i>
SSL_KeyFile <i>file</i>	SSLCertificateKeyFile <i>file</i>
SSL_CipherSuite <i>arg</i>	SSLCipherSuite <i>arg</i>
SSL_X509VerifyDir <i>arg</i>	SSLCACertificatePath <i>arg</i>
SSL_Log <i>file</i>	SSLLogFile <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>	-
Stronghold 2.x compatibility:	
StrongholdAccelerator <i>dir</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
SSLErrorFile <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>	-
SSLProxyMachineCertFile <i>file</i>	-
SSLProxyCACertificatePath <i>dir</i>	-
SSLProxyCACertificateFile <i>file</i>	-
SSLProxyVerifyDepth <i>number</i>	-
SSLProxyCipherList <i>spec</i>	-



When you use ``SSLOptions +CompatEnvVars" additional environment variables are generated. They all correspond to existing official mod_ssl variables. The currently implemented variable derivation is listed 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 built into Apache or at least loaded (under DSO situation) additional functions exist for the [Custom Log Format](#) of [mod_log_config](#) as documented in the Reference Chapter.

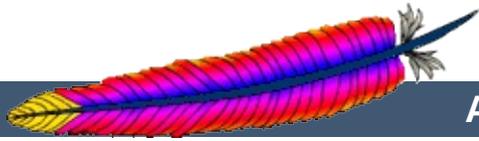
Beside the ``%{varname}x`` eXtension format function which can be used to expand any variables provided by any module, an additional Cryptography ``%{name}c`` cryptography format function exists for backward compatibility. The currently implemented function calls are listed in [Table 3](#).

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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [SSL/TLS](#)

SSL/TLS Strong Encryption: How-To

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

-- Standard textbook cookie

How to solve particular security constraints for an SSL-aware webserver is not always obvious because of the coherences between SSL, HTTP and Apache's way of processing requests. This chapter gives instructions on how to solve such 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 coherences.



- [SSLv2 only server](#)
- [strong encryption only server](#)
- [server gated cryptography](#)
- [stronger per-directory requirements](#)

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 strongest ciphers:

httpd.conf

```
SSLProtocol all -SSLv2
SSLCipherSuite HIGH:!aNULL:!MD5
```

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 details you can find in the README.GlobalID document in the mod_ssl distribution. In short: The server has a Global ID server certificate, signed by a special CA certificate from Verisign which enables strong encryption in export browsers. This works as following: The browser connects with an export cipher, the server sends its Global ID certificate, the browser verifies it and

subsequently upgrades the cipher suite before any HTTP communication takes place. The question now is: How can we allow this upgrade, but enforce strong encryption. Or in other words: Browser either have to initially connect with strong encryption or have to upgrade to strong encryption, but are not allowed to keep the export ciphers. The following does the trick:

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 you cannot just use a server-wide [SSLCipherSuite](#) which restricts the ciphers to the strong variants. But `mod_ssl` allows you to reconfigure the cipher suite in per-directory context and automatically forces a renegotiation of the SSL parameters to meet the new configuration. So, the solution is:

```
# 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:!aNULL:!MD5
</Location>
```



- [simple certificate-based client authentication](#)
- [selective certificate-based client authentication](#)
- [particular certificate-based client authentication](#)
- [intranet vs. internet authentication](#)

How can I authenticate clients based on certificates when I know all my clients?

When you know your user community (i.e. a closed user group situation), as it's the case for instance in an Intranet, you can use plain certificate authentication. All you have 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 authenticate my clients for a particular URL based on certificates but still allow arbitrary clients to access the remaining parts of the server?

For this we again use the per-directory reconfiguration feature of [mod_ssl](#):

httpd.conf

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

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

How can I authenticate only particular clients for a some URLs based on certificates but still allow arbitrary clients to access the remaining parts of the server?

The key is to check for various ingredients of the client certificate. Usually this means to check the whole or part of the Distinguished Name (DN) of the Subject. For this two methods exists: The [mod_auth](#) based variant and the [SSLRequire](#) variant. The first method is good when the clients are of totally different type, i.e. when their DNs have no common fields (usually the organisation, etc.). In this case you've to establish a password database containing *all* clients. The second method is better when your clients are all part of a common hierarchy which is encoded into the DN. Then you can match them more easily.

The first method:

httpd.conf

```
SSLVerifyClient      none
<Directory /usr/local/apache2/htdocs/secure/area>

SSLVerifyClient      require
SSLVerifyDepth       5
SSLCACertificateFile conf/ssl.crt/ca.crt
SSLCACertificatePath conf/ssl.crt
SSLOptions            +FakeBasicAuth
SSLRequireSSL
AuthName              "Snake Oil Authentication"
AuthType              Basic
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
```

The second method:

httpd.conf

```
SSLVerifyClient      none  
<Directory /usr/local/apache2/htdocs/secure/area>  
  
    SSLVerifyClient      require  
    SSLVerifyDepth      5  
    SSLCACertificateFile conf/ssl.crt/ca.crt  
    SSLCACertificatePath conf/ssl.crt  
    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 a subarea on the Intranet website for clients coming from the Internet but still allow plain HTTP access for clients on the Intranet?

Let us assume the Intranet can be distinguished through the IP network 192.168.1.0/24 and the subarea on the Intranet website has the URL /subarea. Then configure the following outside your HTTPS virtual host (so it applies to both HTTPS and HTTP):

httpd.conf

```
SSLCACertificateFile 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]

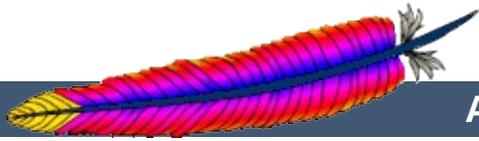
#   Allow Network Access and/or Basic Auth
Satisfy              any

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

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

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 or the mod_ssl Support Mailing List 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 ``, 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 Verisign Global ID 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.certisign.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 host . 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 Verisign Global ID server certificate?

Verisign uses an intermediate CA certificate between the root CA certificate (which is installed in the browsers) and the server certificate (which you installed on the server). You should have received this additional CA certificate from Verisign. If not, complain to them. Then, configure this certificate with the [SSLCertificateChainFile](#) directive. This ensures that the intermediate CA certificate is sent to the browser, filling the gap in the certificate chain.



- [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?](#)
- [Why is it not 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 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. Bingo!

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 comes as rather a shock to learn that it is impossible.

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 does not contain any Host: field, the server has no way to decide which SSL virtual host to use. Usually, it will just use the first one it finds, which matches the port and IP address specified.

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.



- [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/2.0/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

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

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

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.

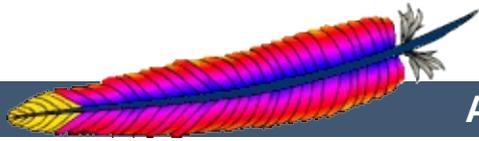
How do I get a backtrace, to help find the reason for my core dump?

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

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

.

.

(authentication)

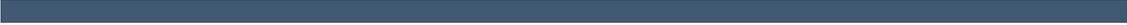
. (authorization)

.



<u>mod_auth</u>	<u>Allow</u>
<u>mod_access</u>	<u>AuthGroupFile</u>
	<u>AuthName</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

```
.  
AuthType Digest .  
mod_auth_digest, . Digest
```

```
AuthName (realm) .
```

```
.  
"Restricted Files" ,  
"Restricted Files" .
```

```
AuthUserFile httpasswd .
```

```
. mod_auth_dbm AuthDBMUserFile . dbmmar
```

```
Require  
require .
```



```
( 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
```

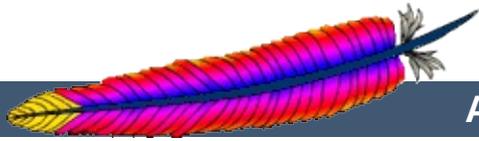
Allow ,



mod_auth mod_access

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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_STRII
GET . FORM METHOD HTML (form)
POST .

. CGI



CGI

Perl CGI

[CPAN](#)

. CGI

CGI.pm.

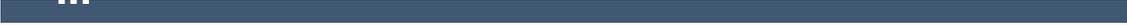
CGI::Lite

C CGI .

<http://www.bout>

CGIC .





CGI . comp.infosystems.www.authoring.cgi

CGI . HTML Writers Guild -servers

<http://www.hwg.org/lists/hwg-servers/>

.

CGI

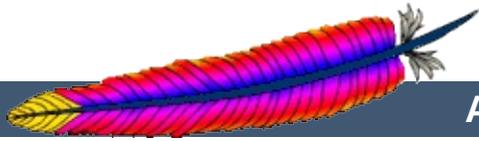
CGI .

[Common Gateway Interface RFC](#) .

CGI

, , CGI

CGI



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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  
.  
XBitHack .  
SSI . , .  
.  
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

,

config element timefmt

attribute .

```
<!--#config timefmt="%A %B %d, %Y" -->  
Today is <!--#echo var="DATE_LOCAL" -->
```

```
<!--#flastmod file="index.html" -->
```

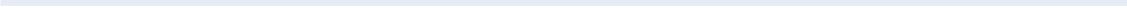
element timefmt .

CGI

SSI , ``

" CGI .

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```



HTML SSI .

?

SSI . . .
HTML . SSI .

```
<!--#config timefmt="%A %B %d, %Y" -->
<!--#flastmod file="ssi.shtml" --> ;
```

ssi.shtml .
LAST_MODIFIED .

```
<!--#config timefmt="%D" -->
This file last modified <!--#echo var="LAST_MODIFIED" -->
```

timefmt strftime . .

(header) (footer) .
include SSI . includ
file attribute virtual attribute . file attribu
. , (/) . ./ .
virtual attribute . / ,

```
<!--#include virtual="/footer.html" -->
```

LAST_MODIFIED .
SSI , .



config()

config() .

SSI

[an error occurred while processing this directive]

config element errmsg attribute .

```
<!--#config errmsg="[It appears that you don't know how to use SSI]" -->
```

SSI

. (?)

sizefmt attribute

config() .

bytes, Kb Mb

abbrev .



```
CGI SSI . e
.SSI ( /bin/sh Win32
DOS) . , .
```

```
<pre>
<!--#exec cmd="ls" -->
</pre>
```

or, on Windows

```
<pre>
<!--#exec cmd="dir" -->
</pre>
```

```
dir ``<dir>" , .
exec . ""
, Options Include
SSI exec .
```



SSI ,

1.2

., 1.2

set

```
<!--#set var="name" value="Rich" -->
```

```
( , LAST_MODIFIED)  
($)
```

```
<!--#set var="modified" value="$LAST_MODIFIED" -->
```

```
<!--#set var="cost" value="\$100" -->
```

```
,  
)
```

```
<!--#set var="date" value="{DATE_LOCAL}_{DATE_GMT}" -->
```

SSI .

if, elif, else, endif .

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

test_condition . , ``"
.(.) , [mod_j](#)
. .

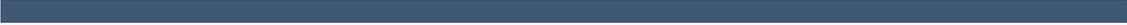
```
BrowserMatchNoCase macintosh Mac
BrowserMatchNoCase MSIE InternetExplorer
```

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

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

IE . JavaScript
IE .
()
CGI .

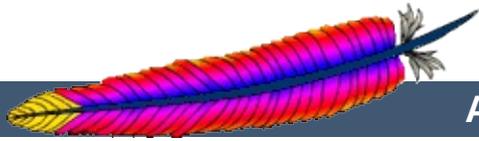




SSI CGI

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

| | [FAQ](#) | |



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

: .htaccess

. .

.htaccess .




```
.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

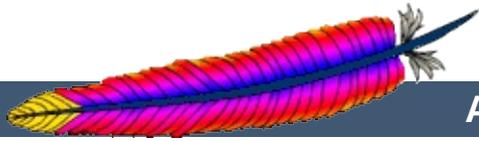
AllowOverride

AllowOverride None

None

.htaccess

.ht



| [FAQ](#) |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 .

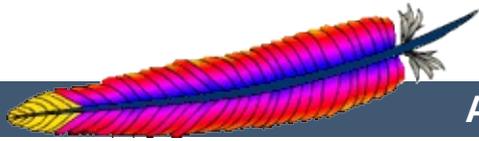
```
http://example.com/~rbowen/cgi-bin/example.cgi
```



,
[AllowOverride](#)

[.htaccess](#)

.htaccess .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

Apache Tutorials

Warning:

This document has not been fully 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.

The following documents give you step-by-step instructions on how to accomplish common tasks with the Apache HTTP server. Many of these documents are located at external sites and are not the work of the Apache Software Foundation. Copyright to documents on external sites is owned by the authors or their assignees. Please consult the [official Apache Server documentation](#) to verify what you read on external sites.



Installation & Getting Started

- [Getting Started with Apache 1.3](#) (ApacheToday)
- [Configuring Your Apache Server Installation](#) (ApacheToday)
- [Getting, Installing, and Running Apache \(on Unix\)](#) (O'Reilly Network Apache DevCenter)
- [Maximum Apache: Getting Started](#) (CNET Builder.com)
- [How to Build the Apache of Your Dreams](#) (Developer Shed)



Basic Configuration

- [An Amble Through Apache Configuration](#) (O'Reilly Network Apache DevCenter)
- [Using .htaccess Files with Apache](#) (ApacheToday)
- [Setting Up Virtual Hosts](#) (ApacheToday)
- [Maximum Apache: Configure Apache](#) (CNET Builder.com)
- [Getting More Out of Apache](#) (Developer Shed)



- [Security and Apache: An Essential Primer](#) (LinuxPlanet)
- [Using User Authentication](#) (Apacheweek)
- [DBM User Authentication](#) (Apacheweek)
- [An Introduction to Securing Apache](#) (Linux.com)
- [Securing Apache - Access Control](#) (Linux.com)
- Apache Authentication [Part 1](#) - [Part 2](#) - [Part 3](#) - [Part 4](#) (ApacheToday)
- [mod_access: Restricting Access by Host](#) (ApacheToday)



Logging

- [Log Rhythms](#) (O'Reilly Network Apache DevCenter)
- [Gathering Visitor Information: Customising Your Logfiles](#) (Apacheweek)
- Apache Guide: Logging [Part 1](#) - [Part 2](#) - [Part 3](#) - [Part 4](#) - [Part 5](#) (ApacheToday)



-
- [Dynamic Content with CGI](#) (ApacheToday)
 - [The Idiot's Guide to Solving Perl CGI Problems](#) (CPAN)
 - [Executing CGI Scripts as Other Users](#) (LinuxPlanet)
 - [CGI Programming FAQ](#) (Web Design Group)
 - Introduction to Server Side Includes [Part 1](#) - [Part 2](#) (ApacheToday)
 - [Advanced SSI Techniques](#) (ApacheToday)
 - [Setting up CGI and SSI with Apache](#) (CNET Builder.com)



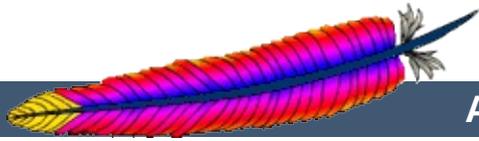
Other Resources

- [Content Negotiation Explained](#) (Apacheweek)
- [Using Apache Imagemaps](#) (Apacheweek)
- [Keeping Your Images from Adorning Other Sites](#) (ApacheToday)
- [Language Negotiation Notes](#) (Alan J. Flavell)

If you have a pointer to an accurate and well-written tutorial not included here, please let us know by submitting it to the [Apache Bug Database](#).

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

Microsoft Windows



Microsoft Windows 2.0 , , .

[Windows](#) () .

Microsoft Windows :

- **Windows NT:** Windows NT Windows . Windows NT, Windows 2000, Windows XP, Windows .Net Server 2003 .
- **Windows 9x:** Windows . Windows 95 (OSR2), Windows 98, Windows ME .



2.0 Windows Windows NT. Intel AM
x86 . Windows 9x

TCP/IP . Windows 95 , Winsock
. Windows 95 Winsock 2 .

Windows NT 4.0 4 TCP/IP Winsock ,
6 .



<http://httpd.apache.org/download.cgi>

. ,
.
Windows .msi Windows .
Microsoft . .zip . Mi
C++ (Visual Studio) .



Microsoft Installer 1.2 . Windows 9x
Installer 2.0 , Windows NT 4.0 2000
. Windows XP .

2.0 . 1.3

. 2.0

.msi . :

1. **(Network Domain).** DNS . ,
DNS server.mydomain.net mydomain.net
2. **(Server Name).** DNS .
server.mydomain.net .
3. **(Administrator's Email Address).**
 .
4. **(For whom to install Apache) 80**
for All Users, on Port 80, as a Service
Recommended (, 80 , service -).
service (,).
only for the Current User, on Por
8080, when started Manually (, 8080 ,
).
5. **(The installation type).**
Typical . Custom .
13 .
6. **(Where to install).** C:\Program
Files\Apache Group, Apache2 .
conf .

```
., .default . ,
conf\httpd.conf conf\httpd.conf.default
. .default , .

, htdocs\index.html
(index.html.default ).,
. , .

conf .

. . .
```



conf . , Windows

Windows :

- Windows , , 2.

MaxRequestsPerChild: ,

MaxRequestsPerChild 0

```
 : . httpd
```

ThreadsPerChild: .

ThreadsPerChild 50.

- Windows .

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

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

- Microsoft IIS Windows ISAPI (Internet Application Programming Interface) (,)
 . [ISAPI](#) .
- CGI [ScriptInterpreterSource](#)
- Windows .htaccess , [AccessFile](#)
- Windows NT Windows error.log . Windows
 Windows NT 4.0 , Windows MMC

Windows 9x Windows .



Windows NT service . Windows

9x

```
service ."
" service " , service ."
. service Administrators
```

Apache Service Monitor .

```
. monitor service service (
)
```

bin

Windows NT service :

```
apache -k install
```

```
service .
```

```
apache -k install -n "MyServiceName"
```

```
service :
```

```
apache -k install -n "MyServiceName" -f "c:\files\my.conf"
```

```
-k install , service Apache2
conf\httpd.conf .
```

```
service .:
```

```
apache -k uninstall
```

```
service :
```

```
apache -k uninstall -n "MyServiceName"
```

service , , Apache Service Monitor NET S
Apache2, NET STOP Apache2 Windows
service :

```
apache -n "MyServiceName" -t
```

service . service :

```
apache -k start
```

service :

```
apache -k stop
```

```
apache -k shutdown
```

service :

```
apache -k restart
```

service (LocalSystem) .
Windows LocalSystem , named pipes, DC
secure RPC .

LocalSystem !

service .

.

1. .

2. Windows NT 4.0 User Manager for Domains , Windows 2000 XP " " " MMC .
3. Users .
4. (htdocs cgi-bin) .
5. logs (RWXD) .
6. Apache.exe (RX) .

```
service (RWXD) log
Apache2 (RX) .
```

" " " " ,
 . service

```
Error code 2186 service ""  

  . , .
```

service Windows Service Control Manager .
 , "" :

```
Could not start the Apache2 service on \\COMPUTER
Error 1067; The process terminated unexpectedly.
```

service .

Windows 9x Windows NT service .

service :

- . ,

```
apache -n "MyServiceName" -k start
```

service . httpd.conf

- Windows 9x NET START NET STOP . service .

- Windows 9x . Windows 9x . Apache Software Foundation Windows 9x .

, , W

Windows NT service , , . , Apache Service Monitor Windows 9x service



```
service . (Windows 9x  
).
```

```
, :
```

```
apache
```

```
Control-C .
```

```
, --> --> Apache HTTP Server 2.0.xx --  
Control Apache Server Start Apache in Console
```

```
. . s€  
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 :
```

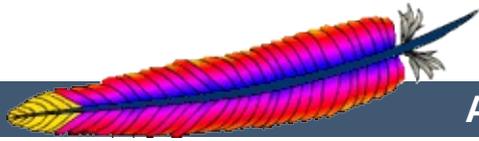
```
http://localhost/
```

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

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

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

```
TCP/IP
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

Microsoft Windows

.

[Microsoft Windows](#)



:

-

50 MB .
MB .

- Microsoft Visual C++ 5.0 .

Visual Studio IDE Workbench .
vcvars32 PATH, INCLUDE, LIB :

```
"c:\Program Files\DevStudio\VC\Bin\vcvars32.bat"
```

- Windows Platform SDK.

Visual C++ 5.0 Microsoft Windo
Platform SDK . setenv Platform

```
"c:\Program Files\Platform SDK\setenv.bat"
```

Visual C++ 6.0 Platform SDK .

```
mod_isapi Windows Platform SDK .  
MSVC++ 5.0 mod_isapi .  
http://msdn.microsoft.com/downloads/sdks/platform/platform.
```

- awk (awk, gawk).

awk.exe
) awk . Brian Kernighan
<http://cm.bell-labs.com/cm/cs/who/bwk/> Win32

<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 .

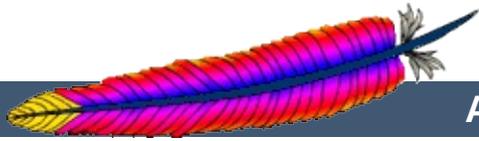
:

- *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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

Novell NetWare

Novell NetWare 6.0 2.0 ,

dev-httpd

[\(FAQ\)](#) ,

, NetWare

novell.devsup.webserver

()

[NetWare](#)

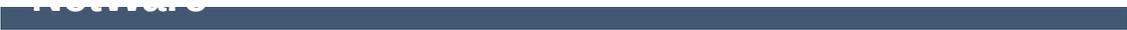


2.0 NetWare 6.0 service pack 3 . SP3 service
pack [NetWare Libraries for C \(LibC\)](#) .

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"

NetWare

DIST

(

[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
```

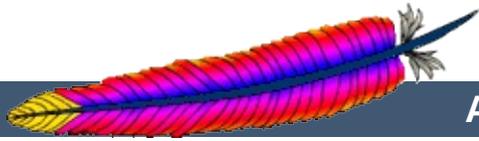
- test_char.h chartables.c \httpd-2.0\os\netware .
- \httpd-2.0 "gmake -f nwgnumakefile"

```
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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

HPUX

Date: Wed, 05 Nov 1997 16:59:34 -0800
From: Rick Jones <raj@cup.hp.com>
Reply-To: 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>
HP-UX 1000 SPECweb96 TIME_WAIT 60
60,000 TCP "" .

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

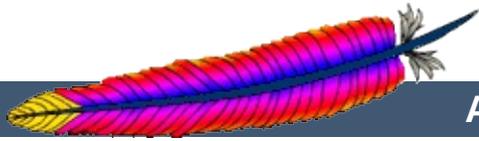
PA-8000 , "chatr".
<> " . GID MLOCK . MLOCK
Setprivgrp(1m) . Glance
.
, mpctl()

.
FIN_WAIT_2 ,
. - 4 .
FIN_WAIT_2 (2)

.
nettune tcp_keepstar
tcp_hash_size ,

, .
,
rick jones

<http://www.cup.hp.com/netperf/NetperfPage.html>



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 .

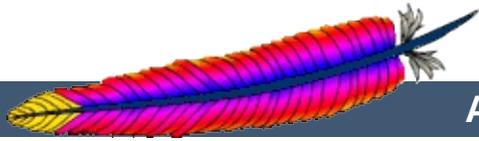


<u>core</u>	+	
<u>mod_access</u>	+	
<u>mod_actions</u>	+	
<u>mod_alias</u>	+	
<u>mod_asis</u>	+	
<u>mod_auth</u>	+	
<u>mod_auth_anon</u>	+	
<u>mod_auth_dbm</u>	?	libdb.a
<u>mod_autoindex</u>	+	
<u>mod_cern_meta</u>	?	
<u>mod_cgi</u>	+	
mod_digest	+	
<u>mod_dir</u>	+	
<u>mod_so</u>	-	
<u>mod_env</u>	+	
<u>mod_example</u>	-	()
<u>mod_expires</u>	+	
<u>mod_headers</u>	+	
<u>mod_imap</u>	+	
<u>mod_include</u>	+	
<u>mod_info</u>	+	
mod_log_agent	+	
<u>mod_log_config</u>	+	
mod_log_referer	+	
<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>	?	



mod_jserv	-	JAVA .
mod_php3	+	mod_php3 LDAP, GD, FreeType .
mod_put	?	
mod_session	-	



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 ]
```

[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
 .

-S
 ().

-t
 . () 0 () 0
 .-D *DUMP_VHOSTS* .

-v
 httpd .

-V
 httpd .

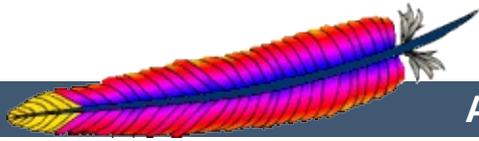
-X
 . , .

[Windows](#) :

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

-n *name*
 name.

-w
 .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 > .

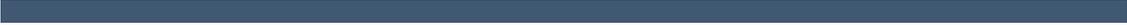
-X *proxy[:port]*

.

-y *<tr>-attributes*
 <tr> .

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



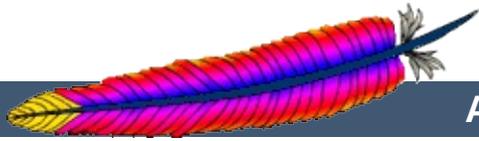


. ' '

.

HTTP/1.x ; " .

; , ab



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 .

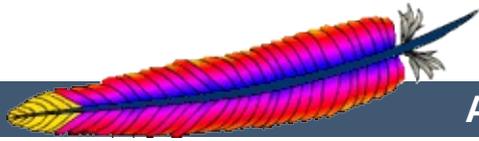
startssl

apachectl -k start -DSSL .

SSL httpd.conf <IfDefine> .

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

|| [FAQ](#) ||



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 *files*
mod_unknown.so .

-D name=value *define .*

-I incdir *include .*

-L libdir

-l libname

-Wc, compiler-flags *compiler-flags .*

-Wl, linker-flags *linker-flags .*

DSO

-i *modules .*

-a *httpd.conf LoadModule*

-A *-a , LoadModule (#) . ,*

-e

. -a -A ,
httpd.conf .

-i



```
mod_foo.c .
```

```
:
```

```
$ apxs -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  
$ _
```

```
LoadModule . apxs  
httpd.conf . :
```

```
$ apxs -i -a mod_foo.c  
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]  
$ _
```

```
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/mod_foo.c  
$ _
```

```
:
```

```
$ cd foo
```

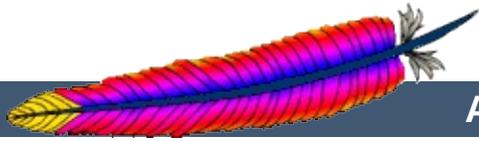
```
$ make all reload
apxs -c mod_foo.c
gcc -fpic -DSHARED_MODULE -I/path/to/apache/include -c
mod_foo.c
ld -Bshareable -o mod_foo.so mod_foo.o
apxs -i -a -n "foo" mod_foo.so
cp mod_foo.so /path/to/apache/modules/mod_foo.so
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module `foo' in /path/to/apache/etc/httpd.conf]
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
$ _
```

apxs PHP3 :

```
$ cd php3
$ ./configure --with-shared-apache=../apache-1.3
$ apxs -c -o libphp3.so mod_php3.c libmodphp3-so.a
gcc -fpic -DSHARED_MODULE -I/tmp/apache/include -c mod_php3.c
ld -Bshareable -o libphp3.so mod_php3.o libmodphp3-so.a
$ _
```

apxs C

. C



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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-imap
    mod_imap 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-auth-ldap
    mod_auth_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 .

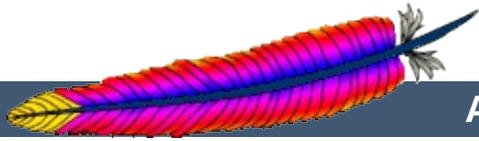
includedir

LDFLAGS

. ,

libdir

-Llibc



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

dbmmanage - DBM

dbmmanage HTTP basic authentication DBM

. dbmmanage .

. [htpasswd](#) .

manpage . [httpd](#)

<http://httpd.apache.org/> .

[httpd](#)

[mod_auth_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

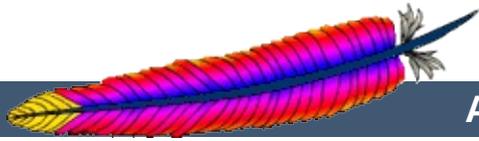
adduser , filename username .

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 ,

.

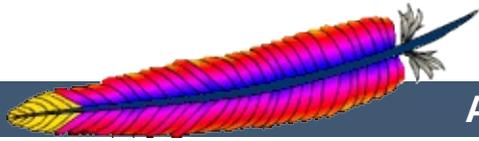
realm

.

username

passwdfile .

username .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 .
```



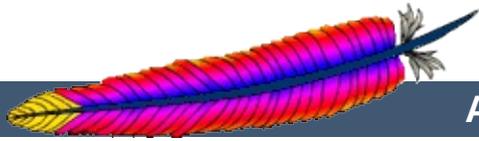
htpasswd

URI

.,

-b .





| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

logresolve - IP-

logresolve IP- .

., IP .

.

IP, .



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



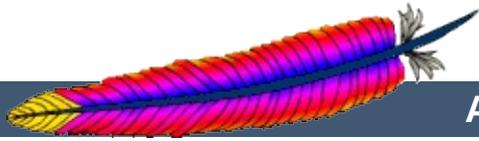
-s *filename*

.

-c

logresolve DNS :IP

IP .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

rotatelogs -

```
rotatelogs
```

```
CustomLog "|bin/rotatelogs /var/logs/logfile 86400" common
```

```
/var/logs/logfile.nnnn . nnnn (
cron ). (24) .
```

```
CustomLog "|bin/rotatelogs /var/logs/logfile 5M" common
```

```
5
```

```
ErrorLog "|bin/rotatelogs /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 5M"
```

```
5
```

```
errorlog.YYYY-mm-dd-
```



```
rotatelogs [ -l ] logfile [ rotationtime [ offset  
] ] | [ filesizeM ]
```



-1 (2.0.51)

GMT . (BST DST) GMT -1
!

logfile

. logfile '%' strftime(3) . '%'
.nnnnnnnnnn .

rotationtime

offset

UTC . 0 UTC . , UTC -5
-300 .

filesizeM

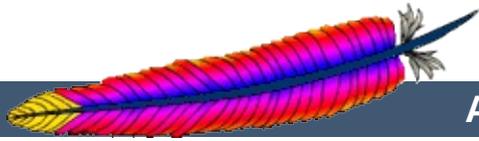
M . rotationtime offset



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	
%%	`%'



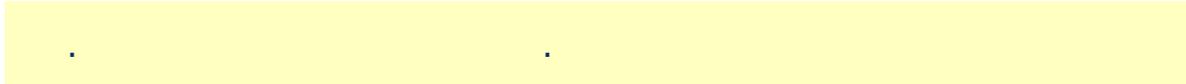
| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

Other Programs



manpage ,



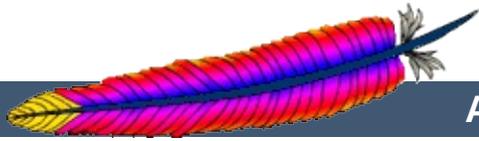
log_server_static

perl cron .



Optimizing the Perl log file

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



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

International Customized Server Error Messages

Warning:

This document has not been fully 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.

This document describes an easy way to provide your Apache HTTP Server with a set of customized error messages which take advantage of [Content Negotiation](#) and `mod_include` to return error messages generated by the server in the client's native language.



By using SSI, all [ErrorDocument](#) messages can share a homogenous and consistent style and layout, and maintenance work (changing images, changing links) is kept to a minimum because all layout information can be kept in a single file.

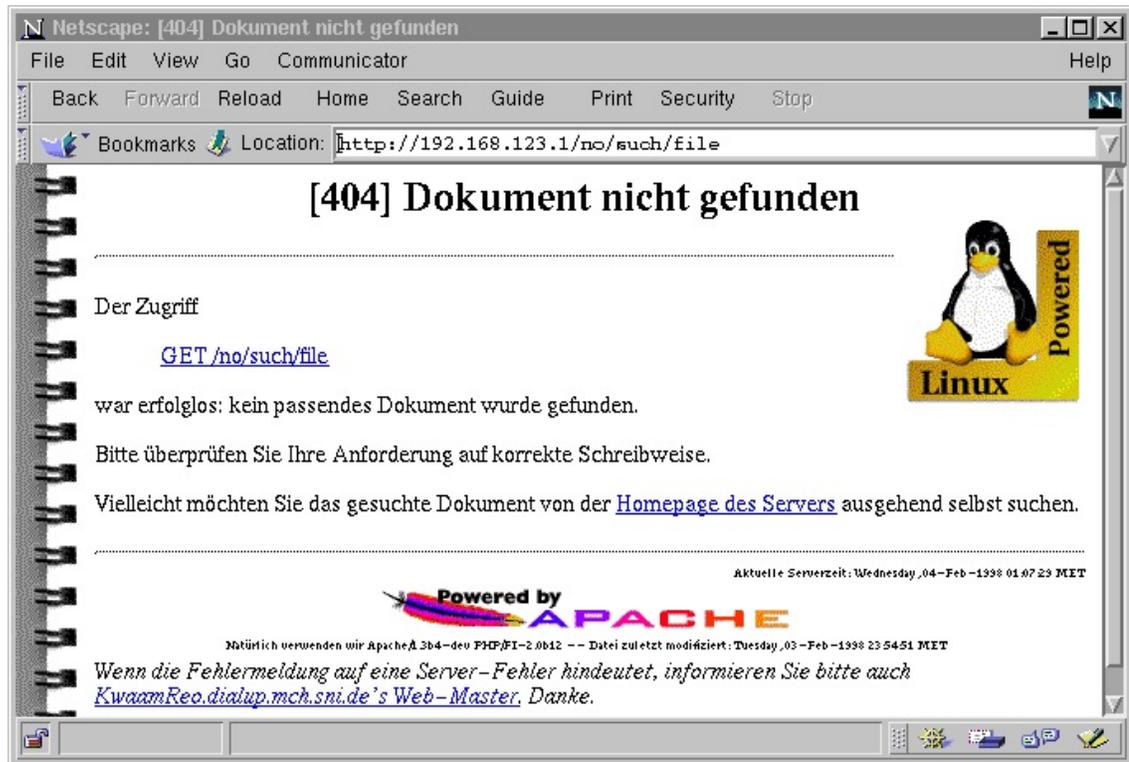
Error documents can be shared across different servers, or even hosts, because all varying information is inserted at the time the error document is returned on behalf of a failed request.

Content Negotiation then selects the appropriate language version of a particular error message text, honoring the language preferences passed in the client's request. (Users usually select their favorite languages in the preferences options menu of today's browsers). When an error document in the client's primary language version is unavailable, the secondary languages are tried or a default (fallback) version is used.

You have full flexibility in designing your error documents to your personal taste (or your company's conventions). For demonstration purposes, we present a simple generic error document scheme. For this hypothetical server, we assume that all error messages...

- possibly are served by different virtual hosts (different host name, different IP address, or different port) on the server machine,
- show a predefined company logo in the right top of the message (selectable by virtual host),
- print the error title first, followed by an explanatory text and (depending on the error context) help on how to resolve the error,
- have some kind of standardized background image,
- display an apache logo and a feedback email address at the bottom of the error message.

An example of a "document not found" message for a german client might look like this:



All links in the document as well as links to the server's administrator mail address, and even the name and port of the serving virtual host are inserted in the error document at "run-time", i.e., when the error actually occurs.



For this concept to work as easily as possible, we must take advantage of as much server support as we can get:

1. By defining the MultiViews [Options](#), we enable the language selection of the most appropriate language alternative (content negotiation).
2. By setting the [LanguagePriority](#) directive we define a set of default fallback languages in the situation where the client's browser did not express any preference at all.
3. By enabling [mod_include](#) (and disallowing execution of cgi scripts for security reasons), we allow the server to include building blocks of the error message, and to substitute the value of certain environment variables into the generated document (dynamic HTML) or even to conditionally include or omit parts of the text.
4. The [AddHandler](#) and [AddType](#) directives are useful for automatically SSI-expanding all files with a `.shtml` suffix to *text/html*.
5. By using the [Alias](#) directive, we keep the error document directory outside of the document tree because it can be regarded more as a server part than part of the document tree.
6. The [<Directory>](#) block restricts these "special" settings to the error document directory and avoids an impact on any of the settings for the regular document tree.
7. For each of the error codes to be handled (see RFC2068 for an exact description of each error code, or look at `src/main/http_protocol.c` if you wish to see apache's standard messages), an [ErrorDocument](#) in the aliased `/error/docs` directory is defined. Note that we only define

the basename of the document here because the MultiViews option will select the best candidate based on the language suffixes and the client's preferences. Any error situation with an error code *not* handled by a custom document will be dealt with by the server in the standard way (*i.e.*, a plain error message in english).

8. Finally, the `AllowOverride` directive tells apache that it is not necessary to look for a `.htaccess` file in the `/error_docs` directory: a minor speed optimization.

The resulting `httpd.conf` configuration would then look similar to this:

Note

Note that you can define your own error messages using this method for only part of the document tree, e.g., a `/~user/` subtree. In this case, the configuration could as well be put into the `.htaccess` file at the root of the subtree, and the `<Directory>` and `</Directory>` directives -but not the contained directives- must be omitted.

```
LanguagePriority en fr de
Alias /error_docs /usr/local/apache/error_docs

<Directory /usr/local/apache/error_docs>
    AllowOverride none
    Options MultiViews IncludesNoExec FollowSymLinks
    AddType text/html .shtml
    <FilesMatch "\.shtml[.]">
        SetOutputFilter INCLUDES
    </FilesMatch>
</Directory>

# "400 Bad Request",
ErrorDocument 400 /error_docs/400
# "401 Authorization Required",
ErrorDocument 401 /error_docs/401
# "403 Forbidden",
ErrorDocument 403 /error_docs/403
```

```
# "404 Not Found",  
ErrorDocument 404 /error_docs/404  
# "500 Internal Server Error",  
ErrorDocument 500 /error_docs/500
```

The directory for the error messages (here: `/usr/local/apache/error_docs/`) must then be created with the appropriate permissions (readable and executable by the server uid or gid, only writable for the administrator).

Naming the Individual Error Document files

By defining the `MULTIViews` option, the server was told to automatically scan the directory for matching variants (looking at language and content type suffixes) when a requested document was not found. In the configuration, we defined the names for the error documents to be just their error number (without any suffix).

The names of the individual error documents are now determined like this (I'm using 403 as an example, think of it as a placeholder for any of the configured error documents):

- No file `error_docs/403` should exist. Otherwise, it would be found and served (with the `DefaultType`, usually `text/plain`), all negotiation would be bypassed.
- For each language for which we have an internationalized version (note that this need not be the same set of languages for each error code - you can get by with a single language version until you actually *have* translated versions), a document `error_docs/403.shtml.lang` is created and filled with the error text in that language ([see below](#)).
- One fallback document called `error_docs/403.shtml` is created, usually by creating a symlink to the default language variant ([see below](#)).

The Common Header and Footer Files

By putting as much layout information in two special "include files", the error documents can be reduced to a bare minimum.

One of these layout files defines the HTML document header and a configurable list of paths to the icons to be shown in the resulting error document. These paths are exported as a set of SSI environment variables and are later evaluated by the "footer" special file. The title of the current error (which is put into the TITLE tag and an H1 header) is simply passed in from the main error document in a variable called `title`.

By changing this file, the layout of all generated error messages can be changed in a second. (By exploiting the features of SSI, you can easily define different layouts based on the current virtual host, or even based on the client's domain name).

The second layout file describes the footer to be displayed at the bottom of every error message. In this example, it shows an apache logo, the current server time, the server version string and adds a mail reference to the site's webmaster.

For simplicity, the header file is simply called `head.shtml` because it contains server-parsed content but no language specific information. The footer file exists once for each language translation, plus a symlink for the default language.

```
for English, French and German versions (default english)

foot.shtml.en,
foot.shtml.fr,
foot.shtml.de,
foot.shtml symlink to
foot.shtml.en
```

Both files are included into the error document by using the directives `<!--#include virtual="head" -->` and `<!--`

`#include virtual="foot" -->` respectively: the rest of the magic occurs in `mod_negotiation` and in `mod_include`.

See [the listings below](#) to see an actual HTML implementation of the discussed example.

Creating ErrorDocuments in Different Languages

After all this preparation work, little remains to be said about the actual documents. They all share a simple common structure:

```
<!--#set var="title" value="error description title" -->
<!--#include virtual="head" -->
    explanatory error text
<!--#include virtual="foot" -->
```

In the [listings section](#), you can see an example of a [400 Bad Request] error document. Documents as simple as that certainly cause no problems to translate or expand.

The Fallback Language

Do we need a special handling for languages other than those we have translations for? We did set the `LanguagePriority`, didn't we?!

Well, the `LanguagePriority` directive is for the case where the client does not express any language priority at all. But what happens in the situation where the client wants one of the languages we do not have, and none of those we do have?

Without doing anything, the Apache server will usually return a [406 no acceptable variant] error, listing the choices from which the client may select. But we're in an error message already, and important error information might get lost when the client had to choose a language representation first.

So, in this situation it appears to be easier to define a fallback

language (by copying or linking, e.g., the english version to a language-less version). Because the negotiation algorithm prefers "more specialized" variants over "more generic" variants, these generic alternatives will only be chosen when the normal negotiation did not succeed.

A simple shell script to do it (execute within the errordocs/ dir):

```
for f in *.shtml.en
do
  ln -s $f `basename $f .en`
done
```



Customizing Proxy Error Messages

As of Apache-1.3, it is possible to use the ErrorDocument mechanism for proxy error messages as well (previous versions always returned fixed predefined error messages).

Most proxy errors return an error code of [500 Internal Server Error]. To find out whether a particular error document was invoked on behalf of a proxy error or because of some other server error, and what the reason for the failure was, you can check the contents of the new ERROR_NOTES CGI environment variable: if invoked for a proxy error, this variable will contain the actual proxy error message text in HTML form.

The following excerpt demonstrates how to exploit the ERROR_NOTES variable within an error document:

```
<!--#if expr="$REDIRECT_ERROR_NOTES = ''" -->

<p>
  The server encountered an unexpected condition
  which prevented it from fulfilling the request.
</p>

<p>
  <a href="mailto:<!--#echo var="SERVER_ADMIN" -->"
  SUBJECT="Error message [<!--#echo var="REDIRECT_STATUS" -->]
  <!--#echo var="title" --> for <!--#echo var="REQUEST_URI" --
  >">
  Please forward this error screen to <!--#echo
  var="SERVER_NAME" -->'s
  WebMaster</a>; it includes useful debugging information
  about
  the Request which caused the error.

  <pre><!--#printenv --></pre>
</p>

<!--#else -->
  <!--#echo var="REDIRECT_ERROR_NOTES" -->

<!--#endif -->
```



So, to summarize our example, here's the complete listing of the `400.shtml.en` document. You will notice that it contains almost nothing but the error text (with conditional additions). Starting with this example, you will find it easy to add more error documents, or to translate the error documents to different languages.

```
<!--#set var="title" value="Bad Request"-->
<!--#include virtual="head" -->

<p>
  Your browser sent a request that this server could not
  understand:
  <blockquote>
    <strong><!--#echo var="REQUEST_URI" --></strong>
  </blockquote>

  The request could not be understood by the server due to
  malformed
  syntax. The client should not repeat the request without
  modifications.
</p>

<p>
  <!--#if expr="$HTTP_REFERER != ''" -->
  Please inform the owner of
  <a href="<!--#echo var="HTTP_REFERER" -->">the referring
  page</a> about
  the malformed link.

  <!--#else -->
  Please check your request for typing errors and retry.

  <!--#endif -->
</p>

<!--#include virtual="foot" -->
```

Here is the complete `head.shtml.en` file (the funny line breaks avoid empty lines in the document after SSI processing). Note the configuration section at top. That's where you configure the images and logos as well as the apache documentation directory. Look how this file displays two different logos depending on the

content of the virtual host name (\$SERVER_NAME), and that an animated apache logo is shown if the browser appears to support it (the latter requires server configuration lines of the form

```
BrowserMatch "^Mozilla/[2-4]" anigif
```

for browser types which support animated GIFs).

```
<!--#if expr="$SERVER_NAME = /*\.*\mycompany\.com/" -->
  <!--#set var="IMG_CorpLogo"
  value="http://$SERVER_NAME:$SERVER_PORT/errordocs/CorpLogo.gif"
  -->
  <!--#set var="ALT_CorpLogo" value="Powered by Linux!" -->

<!--#else -->
  <!--#set var="IMG_CorpLogo"
  value="http://$SERVER_NAME:$SERVER_PORT/errordocs/PrivLogo.gif"
  -->
  <!--#set var="ALT_CorpLogo" value="Powered by Linux!" -->
<!--#endif-->

<!--#set var="IMG_BgImage"
value="http://$SERVER_NAME:$SERVER_PORT/errordocs/BgImage.gif"
-->
<!--#set var="DOC_Apache"
value="http://$SERVER_NAME:$SERVER_PORT/Apache/" -->

<!--#if expr="$anigif" -->
  <!--#set var="IMG_Apache"
  value="http://$SERVER_NAME:$SERVER_PORT/icons/apache_anim.gif"
  -->
<!--#else-->
  <!--#set var="IMG_Apache"
  value="http://$SERVER_NAME:$SERVER_PORT/icons/apache_pb.gif"
  -->
<!--#endif-->

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
  <title>
    [<!--#echo var="REDIRECT_STATUS" -->] <!--#echo var="title"
    -->
  </title>
</head>
```

```

<body bgcolor="white" background="<!--#echo var="IMG_BgImage" -
->">
  <h1 align="center">
    [<!--#echo var="REDIRECT_STATUS" -->] <!--#echo var="title"
    -->
    "
      alt="<!--#echo var="ALT_CorpLogo" -->" align="right">
  </h1>

  <hr /> <!--
  ===== -->
  <div>

```

and this is the foot.shtml.en file:

```

</div>
<hr />

<div align="right">
  <small>Local Server time: <!--#echo var="DATE_LOCAL" -->
  </small>
</div>

<div align="center">
  <a href="<!--#echo var="DOC_Apache" -->">
  " border="0"
  align="bottom"
    alt="Powered by <!--#echo var="SERVER_SOFTWARE" -->">
  </a>
  <br />
  <small><!--#set var="var" value="Powered by
  $SERVER_SOFTWARE --
  File last modified on $LAST_MODIFIED" -->
  <!--#echo var="var" --></small>
</div>

<p>If the indicated error looks like a misconfiguration,
please inform
<a href="mailto:<!--#echo var="SERVER_ADMIN" -->"
subject="Feedback about Error message [<!--#echo
var="REDIRECT_STATUS" -->]
<!--#echo var="title" -->, req=<!--#echo var="REQUEST_URI" -
->">
<!--#echo var="SERVER_NAME" -->'s WebMaster</a>.
</p>

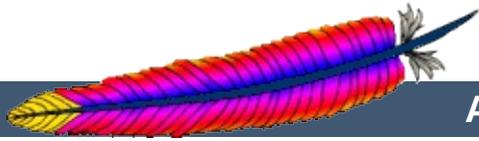
</body>
</html>

```

If you have tips to contribute, send mail to martin@apache.org

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

Connections in the FIN_WAIT_2 state and Apache

Warning:

This document has not been fully 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.

Starting with the Apache 1.2 betas, people are reporting many more connections in the FIN_WAIT_2 state (as reported by `netstat`) than they saw using older versions. When the server closes a TCP connection, it sends a packet with the FIN bit set to the client, which then responds with a packet with the ACK bit set. The client then sends a packet with the FIN bit set to the server, which responds with an ACK and the connection is closed. The state that the connection is in during the period between when the server gets the ACK from the client and the server gets the FIN from the client is known as FIN_WAIT_2. See the [TCP RFC](#) for the technical details of the state transitions.

The FIN_WAIT_2 state is somewhat unusual in that there is no timeout defined in the standard for it. This means that on many operating systems, a connection in the FIN_WAIT_2 state will stay around until the system is rebooted. If the system does not have a timeout and too many FIN_WAIT_2 connections build up, it can fill up the space allocated for storing information about the connections and crash the kernel. The connections in FIN_WAIT_2 do not tie up an `httpd` process.



There are numerous reasons for it happening, some of them may not yet be fully clear. What is known follows.

Buggy Clients and Persistent Connections

Several clients have a bug which pops up when dealing with persistent connections (aka keepalives). When the connection is idle and the server closes the connection (based on the [KeepAliveTimeout](#)), the client is programmed so that the client does not send back a FIN and ACK to the server. This means that the connection stays in the FIN_WAIT_2 state until one of the following happens:

- The client opens a new connection to the same or a different site, which causes it to fully close the older connection on that socket.
- The user exits the client, which on some (most?) clients causes the OS to fully shutdown the connection.
- The FIN_WAIT_2 times out, on servers that have a timeout for this state.

If you are lucky, this means that the buggy client will fully close the connection and release the resources on your server. However, there are some cases where the socket is never fully closed, such as a dialup client disconnecting from their provider before closing the client. In addition, a client might sit idle for days without making another connection, and thus may hold its end of the socket open for days even though it has no further use for it. **This is a bug in the browser or in its operating system's TCP implementation.**

The clients on which this problem has been verified to exist:

- Mozilla/3.01 (X11; I; FreeBSD 2.1.5-RELEASE i386)
- Mozilla/2.02 (X11; I; FreeBSD 2.1.5-RELEASE i386)
- Mozilla/3.01Gold (X11; I; SunOS 5.5 sun4m)

- MSIE 3.01 on the Macintosh
- MSIE 3.01 on Windows 95

This does not appear to be a problem on:

- Mozilla/3.01 (Win95; I)

It is expected that many other clients have the same problem. What a client **should do** is periodically check its open socket(s) to see if they have been closed by the server, and close their side of the connection if the server has closed. This check need only occur once every few seconds, and may even be detected by a OS signal on some systems (e.g., Win95 and NT clients have this capability, but they seem to be ignoring it).

Apache **cannot** avoid these FIN_WAIT_2 states unless it disables persistent connections for the buggy clients, just like we recommend doing for Navigator 2.x clients due to other bugs. However, non-persistent connections increase the total number of connections needed per client and slow retrieval of an image-laden web page. Since non-persistent connections have their own resource consumptions and a short waiting period after each closure, a busy server may need persistence in order to best serve its clients.

As far as we know, the client-caused FIN_WAIT_2 problem is present for all servers that support persistent connections, including Apache 1.1.x and 1.2.

A necessary bit of code introduced in 1.2

While the above bug is a problem, it is not the whole problem. Some users have observed no FIN_WAIT_2 problems with Apache 1.1.x, but with 1.2b enough connections build up in the FIN_WAIT_2 state to crash their server. The most likely source for additional FIN_WAIT_2 states is a function called

`linger_close()` which was added between 1.1 and 1.2. This function is necessary for the proper handling of persistent connections and any request which includes content in the message body (e.g., PUTs and POSTs). What it does is read any data sent by the client for a certain time after the server closes the connection. The exact reasons for doing this are somewhat complicated, but involve what happens if the client is making a request at the same time the server sends a response and closes the connection. Without lingering, the client might be forced to reset its TCP input buffer before it has a chance to read the server's response, and thus understand why the connection has closed. See the [appendix](#) for more details.

The code in `linger_close()` appears to cause problems for a number of factors, including the change in traffic patterns that it causes. The code has been thoroughly reviewed and we are not aware of any bugs in it. It is possible that there is some problem in the BSD TCP stack, aside from the lack of a timeout for the `FIN_WAIT_2` state, exposed by the `linger_close` code that causes the observed problems.



There are several possible workarounds to the problem, some of which work better than others.

Add a timeout for FIN_WAIT_2

The obvious workaround is to simply have a timeout for the FIN_WAIT_2 state. This is not specified by the RFC, and could be claimed to be a violation of the RFC, but it is widely recognized as being necessary. The following systems are known to have a timeout:

- [FreeBSD](#) versions starting at 2.0 or possibly earlier.
- [NetBSD](#) version 1.2(?)
- [OpenBSD](#) all versions(?)
- [BSD/OS](#) 2.1, with the [K210-027](#) patch installed.
- [Solaris](#) as of around version 2.2. The timeout can be tuned by using `ndd` to modify `tcp_fin_wait_2_flush_interval`, but the default should be appropriate for most servers and improper tuning can have negative impacts.
- [Linux](#) 2.0.x and earlier(?)
- [HP-UX](#) 10.x defaults to terminating connections in the FIN_WAIT_2 state after the normal keepalive timeouts. This does not refer to the persistent connection or HTTP keepalive timeouts, but the `SO_LINGER` socket option which is enabled by Apache. This parameter can be adjusted by using `net tune` to modify parameters such as `tcp_keepstart` and `tcp_keeptop`. In later revisions, there is an explicit timer for connections in FIN_WAIT_2 that can be modified; contact HP support for details.
- [SGI IRIX](#) can be patched to support a timeout. For IRIX 5.3, 6.2, and 6.3, use patches 1654, 1703 and 1778 respectively. If you have trouble locating these patches, please contact your SGI support channel for help.
- [NCR's MP RAS Unix](#) 2.xx and 3.xx both have FIN_WAIT_2

timeouts. In 2.xx it is non-tunable at 600 seconds, while in 3.xx it defaults to 600 seconds and is calculated based on the tunable "max keep alive probes" (default of 8) multiplied by the "keep alive interval" (default 75 seconds).

- [Sequent's ptx/TCP/IP for DYNIX/ptx](#) has had a FIN_WAIT_2 timeout since around release 4.1 in mid-1994.

The following systems are known to not have a timeout:

- [SunOS 4.x](#) does not and almost certainly never will have one because it is at the very end of its development cycle for Sun. If you have kernel source should be easy to patch.

There is a [patch available](#) for adding a timeout to the FIN_WAIT_2 state; it was originally intended for BSD/OS, but should be adaptable to most systems using BSD networking code. You need kernel source code to be able to use it.

Compile without using `lingering_close()`

It is possible to compile Apache 1.2 without using the `lingering_close()` function. This will result in that section of code being similar to that which was in 1.1. If you do this, be aware that it can cause problems with PUTs, POSTs and persistent connections, especially if the client uses pipelining. That said, it is no worse than on 1.1, and we understand that keeping your server running is quite important.

To compile without the `lingering_close()` function, add `-DNO_LINGCLOSE` to the end of the `EXTRA_CFLAGS` line in your Configuration file, rerun [Configure](#) and rebuild the server.

Use `SO_LINGER` as an alternative to `lingering_close()`

On most systems, there is an option called `SO_LINGER` that can be set with `setsockopt(2)`. It does something very similar to `linger_close()`, except that it is broken on many systems so that it causes far more problems than `linger_close`. On some systems, it could possibly work better so it may be worth a try if you have no other alternatives.

To try it, add `-DUSE_SO_LINGER -DNO_LINGCLOSE` to the end of the `EXTRA_CFLAGS` line in your Configuration file, rerun [Configure](#) and rebuild the server.

NOTE

Attempting to use `SO_LINGER` and `linger_close()` at the same time is very likely to do very bad things, so don't.

Increase the amount of memory used for storing connection state

BSD based networking code:

BSD stores network data, such as connection states, in something called an mbuf. When you get so many connections that the kernel does not have enough mbufs to put them all in, your kernel will likely crash. You can reduce the effects of the problem by increasing the number of mbufs that are available; this will not prevent the problem, it will just make the server go longer before crashing.

The exact way to increase them may depend on your OS; look for some reference to the number of "mbufs" or "mbuf clusters". On many systems, this can be done by adding the line `NMBCLUSTERS="n"`, where `n` is the number of mbuf clusters you want to your kernel config file and rebuilding your kernel.

Disable KeepAlive

If you are unable to do any of the above then you should, as a last resort, disable KeepAlive. Edit your httpd.conf and change "KeepAlive On" to "KeepAlive Off".



Below is a message from Roy Fielding, one of the authors of HTTP/1.1.

Why the lingering close functionality is necessary with HTTP

The need for a server to linger on a socket after a close is noted a couple times in the HTTP specs, but not explained. This explanation is based on discussions between myself, Henrik Frystyk, Robert S. Thau, Dave Raggett, and John C. Mallery in the hallways of MIT while I was at W3C.

If a server closes the input side of the connection while the client is sending data (or is planning to send data), then the server's TCP stack will signal an RST (reset) back to the client. Upon receipt of the RST, the client will flush its own incoming TCP buffer back to the un-ACKed packet indicated by the RST packet argument. If the server has sent a message, usually an error response, to the client just before the close, and the client receives the RST packet before its application code has read the error message from its incoming TCP buffer and before the server has received the ACK sent by the client upon receipt of that buffer, then the RST will flush the error message before the client application has a chance to see it. The result is that the client is left thinking that the connection failed for no apparent reason.

There are two conditions under which this is likely to occur:

1. sending POST or PUT data without proper authorization
2. sending multiple requests before each response (pipelining) and one of the middle requests resulting in an error or other break-the-connection result.

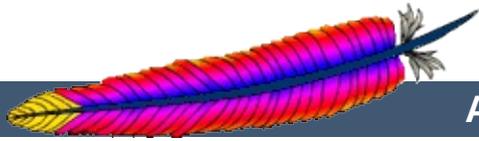
The solution in all cases is to send the response, close only the

write half of the connection (what shutdown is supposed to do), and continue reading on the socket until it is either closed by the client (signifying it has finally read the response) or a timeout occurs. That is what the kernel is supposed to do if `SO_LINGER` is set. Unfortunately, `SO_LINGER` has no effect on some systems; on some other systems, it does not have its own timeout and thus the TCP memory segments just pile-up until the next reboot (planned or not).

Please note that simply removing the linger code will not solve the problem -- it only moves it to a different and much harder one to detect.

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

Known Problems in Clients

Warning:

This document has not been fully 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.

Over time the Apache Group has discovered or been notified of problems with various clients which we have had to work around, or explain. This document describes these problems and the workarounds available. It's not arranged in any particular order. Some familiarity with the standards is assumed, but not necessary.

For brevity, *Navigator* will refer to Netscape's Navigator product (which in later versions was renamed "Communicator" and various other names), and *MSIE* will refer to Microsoft's Internet Explorer product. All trademarks and copyrights belong to their respective companies. We welcome input from the various client authors to correct inconsistencies in this paper, or to provide us with exact version numbers where things are broken/fixed.

For reference, [RFC1945](#) defines HTTP/1.0, and [RFC2068](#) defines HTTP/1.1. Apache as of version 1.2 is an HTTP/1.1 server (with an optional HTTP/1.0 proxy).

Various of these workarounds are triggered by environment variables. The admin typically controls which are set, and for which clients, by using `mod_browser`. Unless otherwise noted all of these workarounds exist in versions 1.2 and later.



Handling CRLF on the server

This is a legacy issue. The CERN webserver required POST data to have an extra CRLF following it. Thus many clients send an extra CRLF that is not included in the Content - Length of the request. Apache works around this problem by eating any empty lines which appear before a request.



Various clients have had broken implementations of *keepalive* (persistent connections). In particular the Windows versions of Navigator 2.0 get very confused when the server times out an idle connection. The workaround is present in the default config files:

```
BrowserMatch Mozilla/2 nokeepalive
```

Note that this matches some earlier versions of MSIE, which began the practice of calling themselves *Mozilla* in their user-agent strings just like Navigator.

MSIE 4.0b2, which claims to support HTTP/1.1, does not properly support *keepalive* when it is used on 301 or 302 (redirect) responses. Unfortunately Apache's `nokeepalive` code prior to 1.2.2 would not work with HTTP/1.1 clients. You must apply [this patch](#) to version 1.2.1. Then add this to your config:

```
BrowserMatch "MSIE 4\.0b2;" nokeepalive
```



To quote from section 3.1 of RFC1945:

HTTP uses a "<MAJOR>.<MINOR>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication.

Since Apache is an HTTP/1.1 server, it indicates so as part of its response. Many client authors mistakenly treat this part of the response as an indication of the protocol that the response is in, and then refuse to accept the response.

The first major indication of this problem was with AOL's proxy servers. When Apache 1.2 went into beta it was the first widespread HTTP/1.1 server. After some discussion, AOL fixed their proxies. In anticipation of similar problems, the `force-response-1.0` environment variable was added to Apache. When present Apache will indicate "HTTP/1.0" in response to an HTTP/1.0 client, but will not in any other way change the response.

The pre-1.1 Java Development Kit (JDK) that is used in many clients (including Navigator 3.x and MSIE 3.x) exhibits this problem. As do some of the early pre-releases of the 1.1 JDK. We think it is fixed in the 1.1 JDK release. In any event the workaround:

```
BrowserMatch Java/1.0 force-response-1.0  
BrowserMatch JDK/1.0 force-response-1.0
```

RealPlayer 4.0 from Progressive Networks also exhibits this problem. However they have fixed it in version 4.01 of the player,

but version 4.01 uses the same User-Agent as version 4.0. The workaround is still:

```
BrowserMatch "RealPlayer 4.0" force-response-1.0
```



HTTP/1.0

MSIE 4.0b2 has this problem. Its Java VM makes requests in HTTP/1.1 format but the responses must be in HTTP/1.0 format (in particular, it does not understand *chunked* responses). The workaround is to fool Apache into believing the request came in HTTP/1.0 format.

```
BrowserMatch "MSIE 4\.0b2;" downgrade-1.0 force-response-1.0
```

This workaround is available in 1.2.2, and in a [patch](#) against 1.2.1.



Security problems with header padding

All versions of Navigator from 2.0 through 4.0b2 (and possibly later) have a problem if the trailing CRLF of the response header starts at offset 256, 257 or 258 of the response. A BrowserMatch for this would match on nearly every hit, so the workaround is enabled automatically on all responses. The workaround implemented detects when this condition would occur in a response and adds extra padding to the header to push the trailing CRLF past offset 258 of the response.



Multipart responses and quoted boundary strings

On multipart responses some clients will not accept quotes (") around the boundary string. The MIME standard recommends that such quotes be used. But the clients were probably written based on one of the examples in RFC2068, which does not include quotes. Apache does not include quotes on its boundary strings to workaround this problem.



byterange request

A byterange request is used when the client wishes to retrieve a portion of an object, not necessarily the entire object. There was a very old draft which included these byteranges in the URL. Old clients such as Navigator 2.0b1 and MSIE 3.0 for the MAC exhibit this behaviour, and it will appear in the servers' access logs as (failed) attempts to retrieve a URL with a trailing ";xxx-yyy". Apache does not attempt to implement this at all.

A subsequent draft of this standard defines a header Request-Range, and a response type multipart/x-byteranges. The HTTP/1.1 standard includes this draft with a few fixes, and it defines the header Range and type multipart/byteranges.

Navigator (versions 2 and 3) sends both Range and Request-Range headers (with the same value), but does not accept a multipart/byteranges response. The response must be multipart/x-byteranges. As a workaround, if Apache receives a Request-Range header it considers it "higher priority" than a Range header and in response uses multipart/x-byteranges.

The Adobe Acrobat Reader plugin makes extensive use of byteranges and prior to version 3.01 supports only the multipart/x-byterange response. Unfortunately there is no clue that it is the plugin making the request. If the plugin is used with Navigator, the above workaround works fine. But if the plugin is used with MSIE 3 (on Windows) the workaround won't work because MSIE 3 doesn't give the Range-Request clue that Navigator does. To workaround this, Apache special cases "MSIE 3" in the User-Agent and serves multipart/x-byteranges. Note that the necessity for this with MSIE 3 is actually due to the Acrobat plugin, not due to the browser.

Netscape Communicator appears to not issue the non-standard Request - Range header. When an Acrobat plugin prior to version 3.01 is used with it, it will not properly understand byteranges. The user must upgrade their Acrobat reader to 3.01.



The HTTP specifications say that it is legal to merge headers with duplicate names into one (separated by commas). Some browsers that support Cookies don't like merged headers and prefer that each Set-Cookie header is sent separately. When parsing the headers returned by a CGI, Apache will explicitly avoid merging any Set-Cookie headers.



Navigator versions 2 through 4 will erroneously re-request GIF89A animations on each loop of the animation if the first response included an Expires header. This happens regardless of how far in the future the expiry time is set. There is no workaround supplied with Apache, however there are hacks for [1.2](#) and for [1.3](#).



In certain situations Navigator 3.01 through 3.03 appear to incorrectly issue a POST without the request body. There is no known workaround. It has been fixed in Navigator 3.04, Netscapes provides some [information](#). There's also [some information](#) about the actual problem.



The http client in the JDK1.2beta2 and beta3 will throw away the first part of the response body when both the headers and the first part of the body are sent in the same network packet AND keep-alive's are being used. If either condition is not met then it works fine.

See also Bug-ID's 4124329 and 4125538 at the java developer connection.

If you are seeing this bug yourself, you can add the following BrowserMatch directive to work around it:

```
BrowserMatch "Java1\.2beta[23]" nokeepalive
```

We don't advocate this though since bending over backwards for beta software is usually not a good idea; ideally it gets fixed, new betas or a final release comes out, and no one uses the broken old software anymore. In theory.



Content-type caching

Navigator (all versions?) will cache the content - type for an object "forever". Using reload or shift-reload will not cause Navigator to notice a content - type change. The only work-around is for the user to flush their caches (memory and disk). By way of an example, some folks may be using an old `mime.types` file which does not map `.htm` to `text/html`, in this case Apache will default to sending `text/plain`. If the user requests the page and it is served as `text/plain`. After the admin fixes the server, the user will have to flush their caches before the object will be shown with the correct `text/html` type.



2000

MSIE versions 3.00 and 3.02 (without the Y2K patch) do not handle cookie expiry dates in the year 2000 properly. Years after 2000 and before 2000 work fine. This is fixed in IE4.01 service pack 1, and in the Y2K patch for IE3.02. Users should avoid using expiry dates in the year 2000.



Lynx incorrectly asking for transparent content negotiation

The Lynx browser versions 2.7 and 2.8 send a "negotiate: trans" header in their requests, which is an indication the browser supports transparent content negotiation (TCN). However the browser does not support TCN. As of version 1.3.4, Apache supports TCN, and this causes problems with these versions of Lynx. As a workaround future versions of Apache will ignore this header when sent by the Lynx client.



MSIE 4.0 does not handle a Vary header properly. The Vary header is generated by mod_rewrite in apache 1.3. The result is an error from MSIE saying it cannot download the requested file. There are more details in [PR#4118](#).

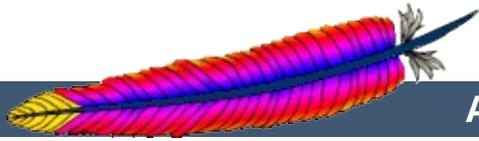
A workaround is to add the following to your server's configuration files:

```
BrowserMatch "MSIE 4\.0" force-no-vary
```

(This workaround is only available with releases **after** 1.3.6 of the Apache Web server.)

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

Descriptors and Apache

Warning:

This document has not been fully 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.

A *descriptor*, also commonly called a *file handle* is an object that a program uses to read or write an open file, or open network socket, or a variety of other devices. It is represented by an integer, and you may be familiar with `stdin`, `stdout`, and `stderr` which are descriptors 0, 1, and 2 respectively. Apache needs a descriptor for each log file, plus one for each network socket that it listens on, plus a handful of others. Libraries that Apache uses may also require descriptors. Normal programs don't open up many descriptors at all, and so there are some latent problems that you may experience should you start running Apache with many descriptors (*i.e.*, with many virtual hosts).

The operating system enforces a limit on the number of descriptors that a program can have open at a time. There are typically three limits involved here. One is a kernel limitation, depending on your operating system you will either be able to tune the number of descriptors available to higher numbers (this is frequently called *FD_SETSIZE*). Or you may be stuck with a (relatively) low amount. The second limit is called the *hard resource* limit, and it is sometimes set by root in an obscure operating system file, but frequently is the same as the kernel limit. The third limit is called the *soft resource* limit. The soft limit is always less than or equal to the hard limit. For example, the hard limit may be 1024, but the soft limit only 64. Any user can raise their soft limit up to the hard limit. Root can raise the hard limit up to the system maximum limit. The soft limit is the actual

limit that is used when enforcing the maximum number of files a process can have open.

To summarize:

```
#open files <= soft limit <= hard limit <= kernel limit
```

You control the hard and soft limits using the `limit` (csh) or `ulimit` (sh) directives. See the respective man pages for more information. For example you can probably use `ulimit -n unlimited` to raise your soft limit up to the hard limit. You should include this command in a shell script which starts your webserver.

Unfortunately, it's not always this simple. As mentioned above, you will probably run into some system limitations that will need to be worked around somehow. Work was done in version 1.2.1 to improve the situation somewhat. Here is a partial list of systems and workarounds (assuming you are using 1.2.1 or later).



Under BSDI 2.0 you can build Apache to support more descriptors by adding `-DFD_SETSIZE=nnn` to `EXTRA_CFLAGS` (where `nnn` is the number of descriptors you wish to support, keep it less than the hard limit). But it will run into trouble if more than approximately 240 Listen directives are used. This may be cured by rebuilding your kernel with a higher `FD_SETSIZE`.



Similar to the BSDI 2.0 case, you should define `FD_SETSIZE` and rebuild. But the extra Listen limitation doesn't exist.



By default Linux has a kernel maximum of 256 open descriptors per process. There are several patches available for the 2.0.x series which raise this to 1024 and beyond, and you can find them in the "unofficial patches" section of [the Linux Information HQ](#). None of these patches are perfect, and an entirely different approach is likely to be taken during the 2.1.x development. Applying these patches will raise the FD_SETSIZE used to compile all programs, and unless you rebuild all your libraries you should avoid running any other program with a soft descriptor limit above 256. As of this writing the patches available for increasing the number of descriptors do not take this into account. On a dedicated webserver you probably won't run into trouble.



CONNECTION LIMIT

Solaris has a kernel hard limit of 1024 (may be lower in earlier versions). But it has a limitation that files using the stdio library cannot have a descriptor above 255. Apache uses the stdio library for the ErrorLog directive. When you have more than approximately 110 virtual hosts (with an error log and an access log each) you will need to build Apache with -
DHIGH_SLACK_LINE=256 added to EXTRA_CFLAGS. You will be limited to approximately 240 error logs if you do this.



AIX version 3.2?? appears to have a hard limit of 128 descriptors.
End of story. Version 4.1.5 has a hard limit of 2000.



Edit the `/etc/conf/cf.d/stune` file or use `/etc/conf/cf.d/configure` choice 7 (User and Group configuration) and modify the `NOFILES` kernel parameter to a suitably higher value. SCO recommends a number between 60 and 11000, the default is 110. Relink and reboot, and the new number of descriptors will be available.



1. Raise `open_max_soft` and `open_max_hard` to 4096 in the `proc` subsystem. Do a `man` on `sysconfig`, `sysconfigdb`, and `sysconfigtab`.
2. Raise `max-vnodes` to a large number which is greater than the number of apache processes * 4096 (Setting it to 250,000 should be good for most people). Do a `man` on `sysconfig`, `sysconfigdb`, and `sysconfigtab`.
3. If you are using Tru64 5.0, 5.0A, or 5.1, define `NO_SLACK` to work around a bug in the OS. `CFLAGS="-DNO_SLACK"`
`./configure`



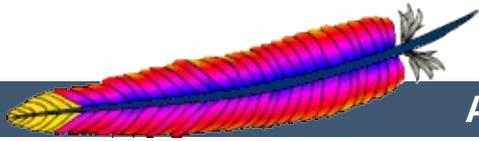
If you have details on another operating system, please submit it through our [Bug Report Page](#).

In addition to the problems described above there are problems with many libraries that Apache uses. The most common example is the bind DNS resolver library that is used by pretty much every unix, which fails if it ends up with a descriptor above 256. We suspect there are other libraries that similar limitations. So the code as of 1.2.1 takes a defensive stance and tries to save descriptors less than 16 for use while processing each request. This is called the *low slack line*.

Note that this shouldn't waste descriptors. If you really are pushing the limits and Apache can't get a descriptor above 16 when it wants it, it will settle for one below 16.

In extreme situations you may want to lower the low slack line, but you shouldn't ever need to. For example, lowering it can increase the limits 240 described above under Solaris and BSDI 2.0. But you'll play a delicate balancing game with the descriptors needed to serve a request. Should you want to play this game, the compile time parameter is `LOW_SLACK_LINE` and there's a tiny bit of documentation in the header file `ht tpd . h`.

Finally, if you suspect that all this slack stuff is causing you problems, you can disable it. Add `-DNO_SLACK` to `EXTRA_CFLAGS` and rebuild. But please report it to our [Bug Report Page](#) so that we can investigate.



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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)
() .

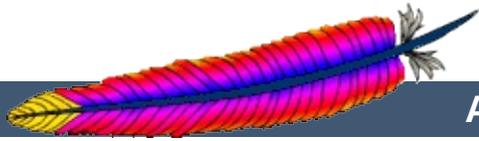
ISO 3166-1

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](#) |



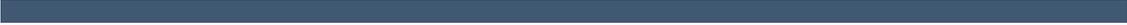
Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



.



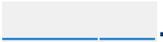


.



.

MPM

"MPM" 

MP

Base

"Base" ,

Extension

"Extension" .

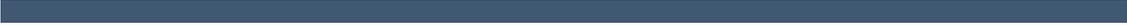
Experimental

"Experimental" ,

External

"External" (" ").

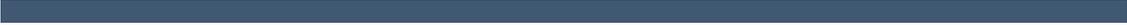




.

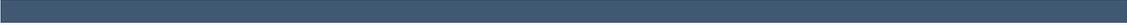
<IfModule> .





, [LoadModule](#) . |
.

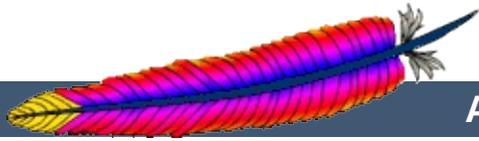




2 ,

• ,

.

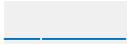


| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



.



(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

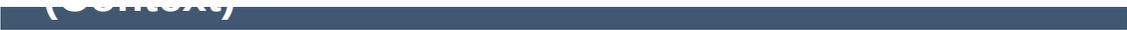


(~~Content~~)

(, .)
"None" .

httpd.conf





. . :

(server config)

(, httpd.conf) , <VirtualHost>
<Directory> . .htaccess .

(virtual host)

<VirtualHost> .

(directory)

Directory, Location, Files ,
<Directory>, <Location>, <Files> .

.htaccess

.htaccess .

.
.
, , .

(boolean) OR . ,

config, .htaccess" httpd.conf .htacc
, <Directory> <VirtualHost> .

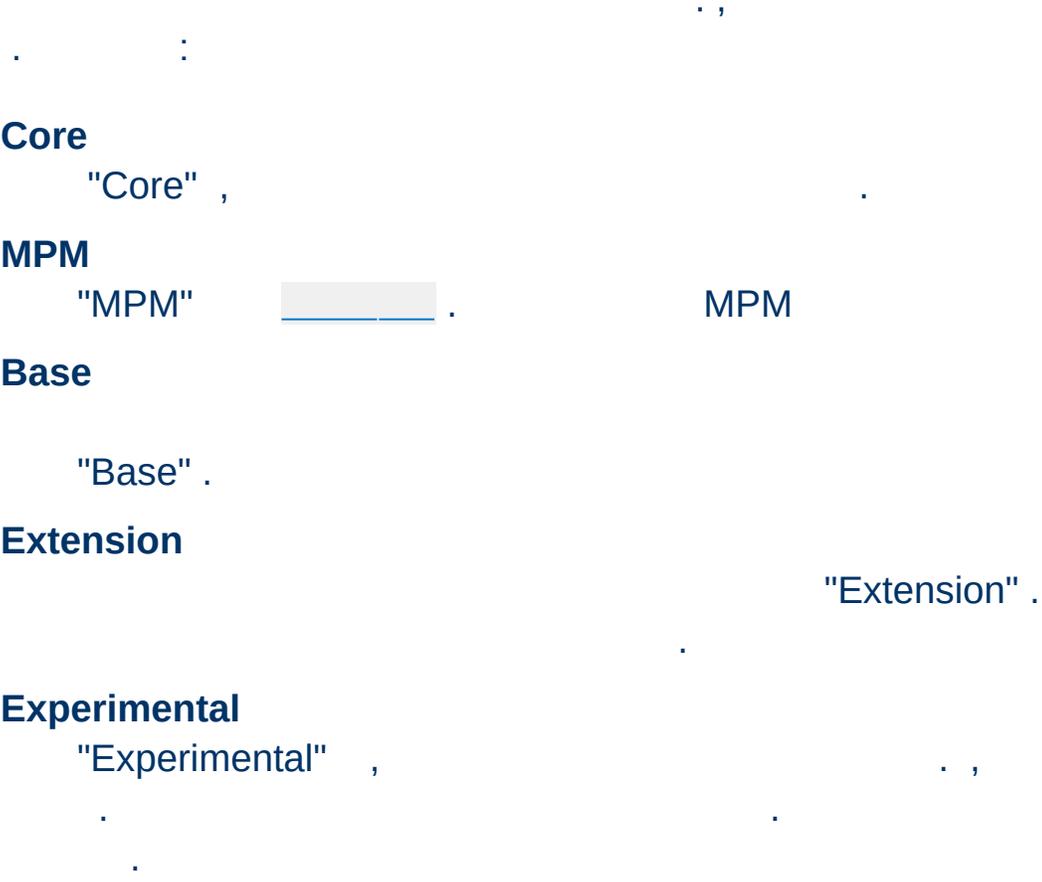
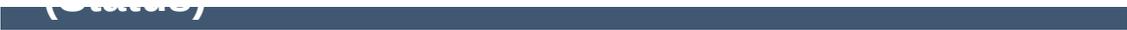


Override (Override)

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

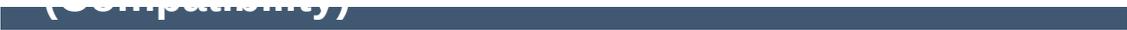
```
Overrides AllowOverride ,      (  
  AllowOverride .  
override .
```





.

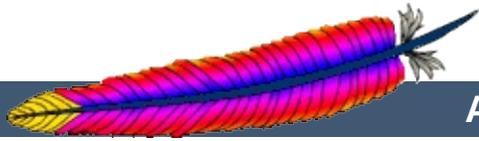




2 ,

.. ,

.



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Core Features

Description: Core Apache HTTP Server features that are always available

Status: Core



Description:	Resources accept trailing pathname information
Syntax:	AcceptPathInfo On Off Default
Default:	AcceptPathInfo Default
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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>
```



Description:	Name of the distributed configuration file
Syntax:	AccessFileName <i>filename</i> [<i>filename</i>] ...
Default:	AccessFileName .htaccess
Context:	server config, virtual host
Status:	Core
Module:	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

Description:	assigns an output filter to a particular MIME-type
Syntax:	AddOutputFilterByType <i>filter</i> [<i>;filter...</i>] <i>MIME-type</i> [<i>MIME-type</i>] ...
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	Available in Apache 2.0.33 and later

This directive activates a particular output [filter](#) for a request depending on the response MIME-type.

The following example uses the DEFLATE filter, which is provided by [mod_deflate](#). It will compress all output (either static or dynamic) which is labeled as text/html or text/plain before it is sent to the client.

```
AddOutputFilterByType DEFLATE text/html text/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.

The by-type output filters are never applied on proxy requests.

See also

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



Description:	Determines whether encoded path separators in URLs are allowed to be passed through
Syntax:	<code>AllowEncodedSlashes On Off</code>
Default:	<code>AllowEncodedSlashes Off</code>
Context:	server config, virtual host
Status:	Core
Module:	core
Compatibility:	Available in Apache 2.0.46 and later

The `AllowEncodedSlashes` directive allows URLs which contain encoded path separators (`%2F` for `/` and additionally `%5C` for `\` on according systems) to be used. Normally such URLs are refused with a 404 (Not found) error.

Turning `AllowEncodedSlashes On` is mostly useful when used in conjunction with `PATH_INFO`.

Note

Allowing encoded slashes does *not* imply *decoding*. Occurrences of `%2F` or `%5C` (*only* on according systems) will be left as such in the otherwise decoded URL string.

See also

- [AcceptPathInfo](#)



Description:	Types of directives that are allowed in <code>.htaccess</code> files
Syntax:	<code>AllowOverride All None directive-type [directive-type] ...</code>
Default:	<code>AllowOverride All</code>
Context:	directory
Status:	Core
Module:	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.).

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

Allow use of the directives controlling specific directory features ([Options](#) and [XBitHack](#)).

Example:

```
AllowOverride AuthConfig Indexes
```

In the example above all directives that are neither in the group `AuthConfig` nor `Indexes` cause an internal server error.

See also

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



Description:	Authorization realm for use in HTTP authentication
Syntax:	AuthName <i>auth-domain</i>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Core
Module:	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](#)



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

This directive selects the type of user authentication for a directory. Only Basic and Digest are currently implemented. It must be accompanied by [AuthName](#) and [Require](#) directives, and directives such as [AuthUserFile](#) and [AuthGroupFile](#) to work.

See also

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



Description:	Technique for locating the interpreter for CGI scripts
Syntax:	<code>CGIMapExtension <i>cgi-path</i> <i>.extension</i></code>
Context:	directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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.



Description:	Enables the generation of Content -MD5 HTTP Response headers
Syntax:	ContentDigest On Off
Default:	ContentDigest Off
Context:	server config, virtual host, directory, .htaccess
Override:	Options
Status:	Core
Module:	core

This directive enables the generation of Content -MD5 headers as defined in RFC1864 respectively RFC2068.

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

Description:	MIME content-type that will be sent if the server cannot determine a type in any other way
Syntax:	DefaultType <i>MIME-type</i>
Default:	DefaultType text/plain
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core

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 must inform the client of the content-type of the document, so in the event of an unknown type it uses the `DefaultType`. For example:

```
DefaultType image/gif
```

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

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.



Description:	Enclose a group of directives that apply only to the named file-system directory and sub-directories
Syntax:	<code><Directory <i>directory-path</i>> ...</code> <code></Directory></code>
Context:	server config, virtual host
Status:	Core
Module:	core

`<Directory>` and `</Directory>` are used to enclose a group of directives that will apply only to the named directory and sub-directories of that directory. 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.

Extended 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 ~ abc$>
  # ... directives here ...
</Directory>
```

the regular expression section won't be considered until after all normal `<Directory>`s and `.htaccess` files have been applied. Then the regular expression will match on `/home/abc/public_html/abc` and the corresponding `<Directory>` will be applied.

Note that the default 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

Description:	Enclose directives that apply to file-system directories matching a regular expression and their subdirectories
Syntax:	<code><DirectoryMatch regex> ... </DirectoryMatch></code>
Context:	server config, virtual host
Status:	Core
Module:	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, 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.

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



Description:	Directory that forms the main document tree visible from the web
Syntax:	DocumentRoot <i>directory-path</i>
Default:	DocumentRoot /usr/local/apache/htdocs
Context:	server config, virtual host
Status:	Core
Module:	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`.

The `DocumentRoot` should be specified without a trailing slash.

See also

- [Mapping URLs to Filesystem Location](#)



Description:	Use memory-mapping to read files during delivery
Syntax:	EnableMMAP On Off
Default:	EnableMMAP On
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	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](#).
- With an NFS-mounted [DocumentRoot](#), the [httpd](#) may crash due to a segmentation fault if a file is deleted or truncated while the [httpd](#) has it memory-mapped.

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>
```



Description:	Use the kernel sendfile support to deliver files to the client
Syntax:	EnableSendfile On Off
Default:	EnableSendfile On
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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.
- 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>
```



Description:	What the server will return to the client in case of an error
Syntax:	ErrorDocument <i>error-code document</i>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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. 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](https://support.microsoft.com/kb/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

Description:	Location where the server will log errors
Syntax:	ErrorLog <i>file-path</i> syslog[: <i>facility</i>]
Default:	ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows and OS/2)
Context:	server config, virtual host
Status:	Core
Module:	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 (|) then it is assumed to be a command to spawn to handle the error log.

Example

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

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

Description:	File attributes used to create the ETag HTTP response header
Syntax:	FileETag <i>component</i> ...
Default:	FileETag INode MTime Size
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	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 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.



Description:	Contains directives that apply to matched filenames
Syntax:	<code><Files filename> ... </Files></code>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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. Extended 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



Description:	Contains directives that apply to regular-expression matched filenames
Syntax:	<code><FilesMatch <i>regex</i>> ... </FilesMatch></code>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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

Description:	Forces all matching files to be served with the specified MIME content-type
Syntax:	ForceType <i>MIME-type</i> None
Context:	directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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>
```



Description:	Enables DNS lookups on client IP addresses
Syntax:	HostnameLookups On Off Double
Default:	HostnameLookups Off
Context:	server config, virtual host, directory
Status:	Core
Module:	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_access](#) 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.



IdentityCheck Directive

Description:	Enables logging of the RFC1413 identity of the remote user
Syntax:	IdentityCheck On Off
Default:	IdentityCheck Off
Context:	server config, virtual host, directory
Status:	Core
Module:	core

This directive enables RFC1413-compliant logging of the remote user name for each connection, where the client machine runs `identd` or something similar. This information is logged in the access log.

The information should not be trusted in any way except for rudimentary usage tracking.

Note that this can cause serious latency problems accessing your server since every request requires one of these lookups to be performed. When firewalls are involved each lookup might possibly fail and add 30 seconds of latency to each hit. So in general this is not very useful on public servers accessible from the Internet.



Description:	Encloses directives that will be processed only if a test is true at startup
Syntax:	<code><IfDefine [!]parameter-name> ... </IfDefine></code>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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 ...
```

```
# httpd.conf
<IfDefine ReverseProxy>
  LoadModule rewrite_module modules/mod_rewrite.so
  LoadModule proxy_module modules/libproxy.so
</IfDefine>
```



Description:	Encloses directives that are processed conditional on the presence or absence of a specific module
Syntax:	<code><IfModule [!]<i>module-name</i>> ...</code> <code></IfModule></code>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	core

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 name*
- *!module name*

In the former case, the directives between the start and end markers are only processed if the module named *module name* 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 name* is **not** included.

The *module name* argument is the file name of the module, at the time it was compiled. For example, `mod_rewrite.c`. 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.



Description:	Includes other configuration files from within the server configuration files
Syntax:	Include <i>file-path directory-path</i>
Context:	server config, virtual host, directory
Status:	Core
Module:	core
Compatibility:	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](#)



Description:	Enables HTTP persistent connections
Syntax:	KeepAlive On Off
Default:	KeepAlive On
Context:	server config, virtual host
Status:	Core
Module:	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.

See also

- [MaxKeepAliveRequests](#)



Description:	Amount of time the server will wait for subsequent requests on a persistent connection
Syntax:	KeepAliveTimeout <i>seconds</i>
Default:	KeepAliveTimeout 15
Context:	server config, virtual host
Status:	Core
Module:	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.



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

Access controls are normally effective for **all** access methods, and this is the usual desired behavior. **In the general case, access control directives should not be placed within a `<Limit>` section.**

The purpose of the `<Limit>` directive is to restrict the effect of the access controls to the nominated HTTP methods. For all other methods, the access restrictions that are enclosed in the `<Limit>` bracket **will have no effect**. The following example applies the access control only to the methods POST, PUT, and DELETE, leaving all other methods unprotected:

```
<Limit POST PUT DELETE>
  Require valid-user
</Limit>
```

The method names listed can be one or more of: GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, and UNLOCK. **The method name is case-sensitive**. If GET is used it will also restrict HEAD requests. The TRACE method cannot be limited.

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.



Description:	Restrict access controls to all HTTP methods except the named ones
Syntax:	<code><LimitExcept <i>method</i> [<i>method</i>] ... ></code> <code>... </LimitExcept></code>
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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>
```



Description:	Determine maximum number of internal redirects and nested subrequests
Syntax:	<code>LimitInternalRecursion <i>number</i> [<i>number</i>]</code>
Default:	<code>LimitInternalRecursion 10</code>
Context:	server config, virtual host
Status:	Core
Module:	core
Compatibility:	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 deep subrequests may be nested. If you specify only one *number*, it will be assigned to both limits.

Example

```
LimitInternalRecursion 5
```



Description:	Restricts the total size of the HTTP request body sent from the client
Syntax:	LimitRequestBody <i>bytes</i>
Default:	LimitRequestBody 0
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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



Description:	Limits the number of HTTP request header fields that will be accepted from the client
Syntax:	<code>LimitRequestFields</code> <i>number</i>
Default:	<code>LimitRequestFields</code> 100
Context:	server config, virtual host
Status:	Core
Module:	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.



Description:	Limits the size of the HTTP request header allowed from the client
Syntax:	<code>LimitRequestFieldsize</code> <i>bytes</i>
Default:	<code>LimitRequestFieldsize</code> 8190
Context:	server config, virtual host
Status:	Core
Module:	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 reduce or increase 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.

Apache 2.0.53 or higher is required for increasing the limit above the compiled-in value of `DEFAULT_LIMIT_REQUEST_FIELDSIZE` (8190 as distributed).

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.



Description:	Limit the size of the HTTP request line that will be accepted from the client
Syntax:	<code>LimitRequestLine bytes</code>
Default:	<code>LimitRequestLine 8190</code>
Context:	server config, virtual host
Status:	Core
Module:	core

This directive sets the number of *bytes* from 0 to the value of the compile-time constant `DEFAULT_LIMIT_REQUEST_LINE` (8190 as distributed) that will be allowed on the HTTP request-line.

The `LimitRequestLine` directive allows the server administrator to reduce the limit on the allowed size of a client's HTTP request-line below the normal input buffer size compiled with the server. 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.



Description:	Limits the size of an XML-based request body
Syntax:	LimitXMLRequestBody <i>bytes</i>
Default:	LimitXMLRequestBody 1000000
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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:	<code><Location URL-path URL> ... </Location></code>
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.

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.

Extended 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 `foo.com`, you might use:

```
<Location /status>
  SetHandler server-status
  Order Deny,Allow
  Deny from all
  Allow from .foo.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



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

Description:	Number of requests allowed on a persistent connection
Syntax:	MaxKeepAliveRequests <i>number</i>
Default:	MaxKeepAliveRequests 100
Context:	server config, virtual host
Status:	Core
Module:	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

Description:	Number of ranges allowed before returning the complete resource
Syntax:	MaxRanges default unlimited none <i>number-of-ranges</i>
Default:	MaxRanges 200
Context:	server config, virtual host, directory
Status:	Core
Module:	core
Compatibility:	Available in Apache HTTP Server 2.0.65 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.



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

```
NameVirtualHost 111.22.33.44
```

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 *

```
NameVirtualHost *
```

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
<VirtualHost 1.2.3.4>
# ...
</VirtualHost>
```

See also

- [Virtual Hosts documentation](#)



Description:	Configures what features are available in a particular directory
Syntax:	Options [+ -] <i>option</i> [[+ -] <i>option</i>] ...
Default:	Options All
Context:	server config, virtual host, directory, .htaccess
Override:	Options
Status:	Core
Module:	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.

Note also, that this option **gets ignored** if set inside a `<Location>` section.

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

This option gets ignored if set inside a [<Location>](#) section.

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**.



Require Directive

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

This directive selects which authenticated users can access a resource. The allowed syntaxes are:

Require user *userid* [*userid*] ...

Only the named users can access the resource.

Require group *group-name* [*group-name*] ...

Only users in the named groups can access the resource.

Require valid-user

All valid users can access the resource.

Require 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.

See also

- [Satisfy](#)
- [mod_access](#)



Description:	Limits the CPU consumption of processes launched by Apache children
Syntax:	<code>RLimitCPU seconds max [seconds max]</code>
Default:	Unset; uses operating system defaults
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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 off from Apache children servicing requests, not the Apache children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked off from the Apache parent such as piped logs.

CPU resource limits are expressed in seconds per process.

See also

- [RLimitMEM](#)
- [RLimitNPROC](#)



Description:	Limits the memory consumption of processes launched by Apache children
Syntax:	RLimitMEM <i>bytes max</i> [<i>bytes max</i>]
Default:	Unset; uses operating system defaults
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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 off from Apache children servicing requests, not the Apache children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked off from the Apache parent such as piped logs.

Memory resource limits are expressed in bytes per process.

See also

- [RLimitCPU](#)
- [RLimitNPROC](#)



Description:	Limits the number of processes that can be launched by processes launched by Apache children
Syntax:	<code>RLimitNPROC <i>number</i> max [<i>number</i> max]</code>
Default:	Unset; uses operating system defaults
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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 off from Apache children servicing requests, not the Apache children themselves. This includes CGI scripts and SSI exec commands, but not any processes forked off 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](#)



Description:	Interaction between host-level access control and user authentication
Syntax:	Satisfy Any All
Default:	Satisfy All
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Core
Module:	core
Compatibility:	Influenced by <Limit> and <LimitExcept> in version 2.0.51 and later

Access policy if both [Allow](#) and [Require](#) used. The parameter can be either All or Any. This directive is only useful if access to a particular area is being restricted by both username/password *and* client host address. In this case the default behavior (All) is to require that the client passes the address access restriction *and* enters a valid username and password. With the Any option the client will be granted access if they either pass the host restriction or enter a valid username and password. This can be used to password restrict an area, but to let clients from particular addresses in without prompting for a password.

For example, if you wanted to let people on your network have unrestricted access to a portion of your website, but require that people outside of your network provide a password, you could use a configuration similar to the following:

```
Require valid-user
Allow from 192.168.1
Satisfy Any
```

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

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

See also

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



Description:	Technique for locating the interpreter for CGI scripts
Syntax:	ScriptInterpreterSource Registry Registry-Strict Script
Default:	ScriptInterpreterSource Script
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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.

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 <i>email-address</i>
Context:	server config, virtual host
Status:	Core
Module:	core

The `ServerAdmin` sets the e-mail address that the server includes in any error messages it returns to the client.

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](#).

```
<VirtualHost *>
ServerName server.domain.com
ServerAlias server server2.domain.com server2
# ...
</VirtualHost>
```

See also

- [Apache Virtual Host documentation](#)



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

The `ServerName` directive sets the hostname and port that the server uses to identify itself. This is used when creating redirection URLs. 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:80
```

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.

See the description of the [UseCanonicalName](#) directive for

settings which determine whether self-referential URL's (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](#)
- [NameVirtualHost](#)
- [ServerAlias](#)



Description:	Legacy URL pathname for a name-based virtual host that is accessed by an incompatible browser
Syntax:	<code>ServerPath</code> <i>URL-path</i>
Context:	virtual host
Status:	Core
Module:	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](#)



Description:	Base directory for the server installation
Syntax:	<code>ServerRoot</code> <i>directory-path</i>
Default:	<code>ServerRoot</code> <code>/usr/local/apache</code>
Context:	server config
Status:	Core
Module:	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`



Description:	Configures the footer on server-generated documents
Syntax:	ServerSignature On Off EMail
Default:	ServerSignature Off
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	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](#)



Description:	Configures the Server HTTP response header
Syntax:	ServerTokens Major Minor Min[imal] Prod[uctOnly] OS
Default:	ServerTokens Full
Context:	server config
Status:	Core
Module:	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](#)



Description:	Forces all matching files to be processed by a handler
Syntax:	SetHandler <i>handler-name</i> None
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	core
Compatibility:	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](#)



Description:	Sets the filters that will process client requests and POST input
Syntax:	<code>SetInputFilter <i>filter</i> [<i>;filter...</i>]</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	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



Description:	Sets the filters that will process responses from the server
Syntax:	<code>SetOutputFilter <i>filter</i> [<i>;filter...</i>]</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Core
Module:	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



Description:	Amount of time the server will wait for certain events before failing a request
Syntax:	<code>TimeOut <i>seconds</i></code>
Default:	<code>TimeOut 300</code>
Context:	server config, virtual host
Status:	Core
Module:	core

The `TimeOut` directive currently defines the amount of time Apache will wait for three things:

1. The total amount of time it takes to receive a GET request.
2. The amount of time between receipt of TCP packets on a POST or PUT request.
3. The amount of time between ACKs on transmissions of TCP packets in responses.

We plan on making these separately configurable at some point down the road. The timer used to default to 1200 before 1.2, but has been lowered to 300 which is still far more than necessary in most situations. It is not set any lower by default because there may still be odd places in the code where the timer is not reset when a packet is sent.



TRACE_MODULE DIRECTIVE

Description:	Determines the behaviour on TRACE requests
Syntax:	TraceEnable <i>[on off extended]</i>
Default:	TraceEnable on
Context:	server config
Status:	Core
Module:	core
Compatibility:	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 On
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

- [ServerName](#)
- [Listen](#)



Description:	Contains directives that apply only to a specific hostname or IP address
Syntax:	<code><VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost></code>
Context:	server config
Status:	Core
Module:	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;
- 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>  
  ServerAdmin webmaster@host.foo.com  
  DocumentRoot /www/docs/host.foo.com  
  ServerName host.foo.com  
  ErrorLog logs/host.foo.com-error_log  
  TransferLog logs/host.foo.com-access_log  
</VirtualHost>
```

IPv6 addresses must be specified in square brackets because the optional port number could not be determined otherwise. An IPv6

example is shown below:

```
<VirtualHost [2001:db8::a00:20ff:fea7:ccea]>
  ServerAdmin webmaster@host.example.com
  DocumentRoot /www/docs/host.example.com
  ServerName host.example.com
  ErrorLog logs/host.example.com-error_log
  TransferLog logs/host.example.com-access_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_`.)

Security

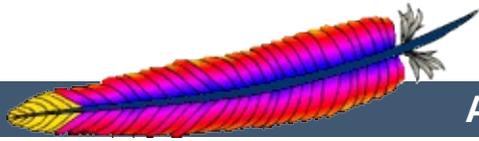
See the [security tips](#) document for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server.

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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM Common Directives

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

Status: MPM



Description:	Method that Apache uses to serialize multiple children accepting requests on network sockets
Syntax:	AcceptMutex Default <i>method</i>
Default:	AcceptMutex Default
Context:	server config
Status:	MPM
Module:	leader , perchild , prefork , threadpool , worker

The `AcceptMutex` directives 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](#).



Description:	Define the non-privileged account on BS2000 machines
Syntax:	BS2000Account <i>account</i>
Context:	server config
Status:	MPM
Module:	perchild , prefork
Compatibility:	Only available for BS2000 machines

The `BS2000Account` directive is available for BS2000 hosts only. It must be used to define the account number for the non-privileged apache server user (which was configured using the `User` directive). This is required by the BS2000 POSIX subsystem (to change the underlying BS2000 task environment by performing a sub-LOGON) to prevent CGI scripts from accessing resources of the privileged account which started the server, usually SYSROOT.

Note

Only one `BS2000Account` directive can be used.

See also

- [Apache EBCDIC port](#)



CoreDumpDirectory Directive

Description:	Directory where Apache attempts to switch before dumping core
Syntax:	CoreDumpDirectory <i>directory</i>
Default:	See usage for the default setting
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_winnt , perchild , prefork , threadpool , worker

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.



Description:	Enables a hook that runs exception handlers after a crash
Syntax:	EnableExceptionHook On Off
Default:	EnableExceptionHook Off
Context:	server config
Status:	MPM
Module:	leader , perchild , prefork , threadpool , worker
Compatibility:	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.



Description:	Group under which the server will answer requests
Syntax:	Group <i>unix-group</i>
Default:	Group #-1
Context:	server config
Status:	MPM
Module:	beos , leader , mpmt_os2 , perchild , prefork , threadpool , worker
Compatibility:	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.



Description:	IP addresses and ports that the server listens to
Syntax:	<code>Listen [IP-address:]portnumber</code>
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_netware , mpm_winnt , mpmt_os2 , perchild , prefork , threadpool , worker
Compatibility:	Required directive since Apache 2.0

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
```

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](#)



Description:	Maximum length of the queue of pending connections
Syntax:	ListenBacklog <i>backlog</i>
Default:	ListenBacklog 511
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_network , mpm_winnt , mpmt_os2 , perchild , prefork , threadpool , worker

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.



Description:	Location of the accept serialization lock file
Syntax:	LockFile <i>filename</i>
Default:	LockFile logs/accept.lock
Context:	server config
Status:	MPM
Module:	leader , perchild , prefork , threadpool , worker

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](#)



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

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`.



Description:	Maximum amount of memory that the main allocator is allowed to hold without calling <code>free()</code>
Syntax:	<code>MaxMemFree</code> <i>KBytes</i>
Default:	<code>MaxMemFree</code> 0
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_network , prefork , threadpool , worker , mpm_winnt

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

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

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.



MaxSpareThreads Directive

Description:	Maximum number of idle threads
Syntax:	MaxSpareThreads <i>number</i>
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_netware , mpmt_os2 , perchild , threadpool , worker

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

For [perchild](#) the default is MaxSpareThreads 10. This MPM monitors the number of idle threads on a per-child basis. If there are too many idle threads in that child, the server will begin to kill threads within that child.

For [worker](#), [leader](#) and [threadpool](#) 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:

- `perchild` requires `MaxSpareThreads` to be less or equal than `ThreadLimit`.
- `mpm_netware` wants the value to be greater than `MinSpareThreads`.
- For `leader`, `threadpool` and `worker` the value must be greater or equal than the sum of `MinSpareThreads` and `ThreadsPerChild`.

See also

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



Description:	Minimum number of idle threads available to handle request spikes
Syntax:	MinSpareThreads <i>number</i>
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_netware , mpmt_os2 , perchild , threadpool , worker

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

[perchild](#) uses a default of `MinSpareThreads 5` and monitors the number of idle threads on a per-child basis. If there aren't enough idle threads in that child, the server will begin to create new threads within that child. Thus, if you set `NumServers` to 10 and a `MinSpareThreads` value of 5, you'll have at least 50 idle threads on your system.

[worker](#), [leader](#) and [threadpool](#) use a default of `MinSpareThreads 75` and deal with idle threads on a server-wide basis. If there aren't enough idle threads in the server then child processes are created until the number of idle threads is greater than number.

[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](#)



Description:	File where the server records the process ID of the daemon
Syntax:	<code>PidFile filename</code>
Default:	<code>PidFile logs/httpd.pid</code>
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_winnt , mpmt_os2 , perchild , prefork , threadpool , worker

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.



Description:	TCP receive buffer size
Syntax:	ReceiveBufferSize <i>bytes</i>
Default:	ReceiveBufferSize 0
Context:	server config
Status:	MPM
Module:	<u>beos</u> , <u>leader</u> , <u>mpm_network</u> , <u>mpm_winnt</u> , <u>mpmt_os2</u> , <u>perchild</u> , <u>prefork</u> , <u>threadpool</u> , <u>worker</u>

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.



Description:	Location of the file used to store coordination data for the child processes
Syntax:	ScoreBoardFile <i>file-path</i>
Default:	ScoreBoardFile logs/apache_status
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_winnt , perchild , prefork , threadpool , worker

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_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](#)



Description:	TCP buffer size
Syntax:	SendBufferSize <i>bytes</i>
Default:	SendBufferSize 0
Context:	server config
Status:	MPM
Module:	beos , leader , mpm_network , mpm_winnt , mpmt_os2 , perchild , prefork , threadpool , worker

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.



Description:	Upper limit on configurable number of processes
Syntax:	ServerLimit <i>number</i>
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	leader , perchild , prefork , threadpool , worker

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](#), [leader](#) and [threadpool](#) 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](#).

With the [perchild](#) MPM, use this directive only if you need to set [NumServers](#) higher than 8 (default).

Note

There is a hard limit of `ServerLimit 20000` compiled into the server. This is intended to avoid nasty effects caused by typos.

See also

- [Stopping and Restarting Apache](#)



Description:	Number of child server processes created at startup
Syntax:	StartServers <i>number</i>
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	leader , mpmt_os2 , prefork , threadpool , worker

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 [leader](#), [threadpool](#) and [worker](#) the default is `StartServers 3`. For [prefork](#) defaults to 5 and for [mpmt_os2](#) to 2.



STARTTHREADS DIRECTIVE

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

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 [perchild](#) the default is `StartThreads 5` and this directive tracks the number of threads per process at startup.

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.



Description:	Sets the upper limit on the configurable number of threads per child process
Syntax:	ThreadLimit <i>number</i>
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	leader , mpm_winnt , perchild , threadpool , worker
Compatibility:	Available for mpm_winnt 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.



Description:	Number of threads created by each child process
Syntax:	ThreadsPerChild <i>number</i>
Default:	See usage for details
Context:	server config
Status:	MPM
Module:	leader , mpm_winnt , threadpool , worker

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.



Description:	The userid under which the server will answer requests
Syntax:	User <i>unix-userid</i>
Default:	User #-1
Context:	server config
Status:	MPM
Module:	leader , perchild , prefork , threadpool , worker
Compatibility:	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

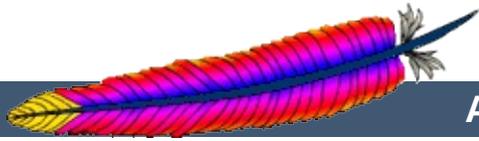
Don't set `User` (or `Group`) to root unless you know exactly what you are doing, and what the dangers are.

With the `perchild` MPM, which is intended to server virtual hosts run under different user IDs, the `User` directive defines the user ID for the main server and the fallback for `<VirtualHost>` sections without an `AssignUserID` directive.

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 `mpm_t_os2` MPMs, it is actually a no-op there and only exists for compatibility reasons.



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

MPM beos

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

(MPM) BeOS .

[\[redacted\]](#)



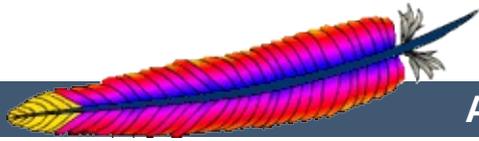
MaxRequestsPerThread

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

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

- () (memory leakage) ;
- .

```
KeepAlive .
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

MPM leader

```
└─ worker MPM
└─ MPM
└─ mpm_leader_module
└─ leader.c
```

MPM , .

worker MPM . Leader/Followers

<http://deuce.doc.wustl.edu/doc/pspdfs/lf.pdf> .

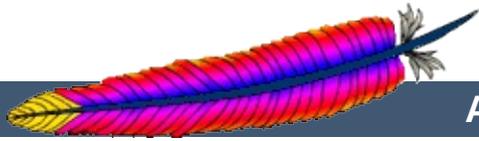
leader MPM , httpd configure --v
mpm=leader .

MPM APR atomic compare-and-swap . x86

386 , SPARC UltraSPARC

--enable-nonportable-atomics=yes . A

CPU atomic .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM netware

Description:	Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare
Status:	MPM
Module Identifier:	mpm_netware_module
Source File:	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

Description:	Set the maximum number of worker threads
Syntax:	MaxThreads <i>number</i>
Default:	MaxThreads 2048
Context:	server config
Status:	MPM
Module:	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
```

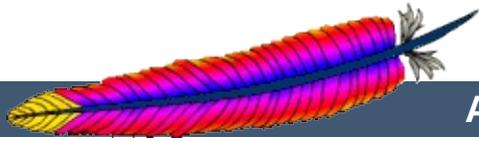


Description:	Determine the stack size for each thread
Syntax:	ThreadStackSize <i>number</i>
Default:	ThreadStackSize 65536
Context:	server config
Status:	MPM
Module:	mpm_netware

This directive tells the server what stack size to use for each of the running threads. If you ever get a stack overflow you will need to bump this number to a higher setting.

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM os2

Description:	Hybrid multi-process, multi-threaded MPM for OS/2
Status:	MPM
Module Identifier:	mpm_mpmt_os2_module
Source File:	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.

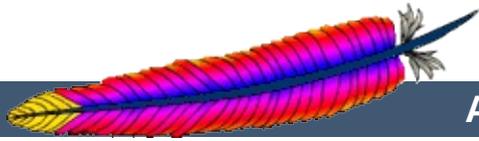
Each child process consists of a a pool of worker threads and a main thread that accepts connections and passes them to the workers via a work queue. The worker thread pool is dynamic, managed by a maintenance thread so that the number of idle threads is kept between [MinSpareThreads](#) and [MaxSpareThreads](#).

See also

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

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM perchild

Description:	Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids
Status:	MPM
Module Identifier:	mpm_perchild_module
Source File:	perchild.c

Summary

This module is not functional. Development of this module is not complete and is not currently active. Do not use [perchild](#) unless you are a programmer willing to help fix it.

This Multi-Processing Module (MPM) implements a hybrid multi-process, multi-threaded web server. A fixed number of processes create threads to handle requests. Fluctuations in load are handled by increasing or decreasing the number of threads in each process.

See also

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



A single control process launches the number of child processes indicated by the [NumServers](#) directive at server startup. Each child process creates threads as specified in the [StartThreads](#) directive. The individual threads then listen for connections and serve them 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 new threads to be created. For each child process, Apache assesses the number of idle threads and creates or destroys threads 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 is determined by multiplying the number of server processes that will be created ([NumServers](#)) by the maximum number of threads created in each process ([MaxThreadsPerChild](#)).

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.

Working with different user-IDs

The `perchild` MPM adds the extra ability to specify that particular processes should serve requests under different user-IDs. These user-IDs can then be associated with specific virtual hosts. You have to use one `ChildPerUserID` directive for every user/group combination you want to be run. Then you can tie particular virtual hosts to that user and group IDs.

The following example runs 7 child processes. Two of them are run under user1/group1. The next four are run under user2/group2 and the remaining process uses the `User` and `Group` of the main server:

Global config

```
NumServers 7
ChildPerUserID user1 group1 2
ChildPerUserID user2 group2 4
```

Using unbalanced numbers of processes as above is useful, if the particular virtual hosts produce different load. The assignment to the virtual hosts is easily done as in the example below. In conclusion with the example above the following assumes, that server2 has to serve about twice of the hits of server1.

Example

```
NameVirtualHost *

<VirtualHost *>
    ServerName fallbackhost
    # no assignment; use fallback
</VirtualHost>

<VirtualHost *>
    ServerName server1
    AssignUserID user1 group1
</VirtualHost>
```

```
<VirtualHost *>  
  ServerName server2  
  AssignUserID user2 group2  
</VirtualHost>
```



Description:	Tie a virtual host to a user and group ID
Syntax:	AssignUserID <i>user-id group-id</i>
Context:	virtual host
Status:	MPM
Module:	perchild

Tie a virtual host to a specific user/group combination. Requests addressed to the virtual host where this directive appears will be served by a process running with the specified user and group ID.

The user and group ID has to be assigned to a number of children in the global server config using the [ChildPerUserID](#) directive. See the section above for a [configuration example](#).



Description:	Specify user ID and group ID for a number of child processes
Syntax:	<code>ChildPerUserID <i>user-id group-id num-children</i></code>
Context:	server config
Status:	MPM
Module:	perchild

Specify a user ID and group ID for a number of child processes. The third argument, *num-children*, is the number of child processes to start with the specified user and group. It does *not* represent a specific child number. 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.

If the total number of child processes, found by totaling all of the third arguments to all `ChildPerUserID` directives in the config file, is less than `NumServers`, then all remaining children will inherit the `User` and `Group` settings from the main server. See the section above for a [configuration example](#).

Security

Don't set *user-id* (or *group-id*) to root unless you know exactly what you are doing, and what the dangers are.



MAXTHREADSPERCHILD DIRECTIVE

Description:	Maximum number of threads per child process
Syntax:	MaxThreadsPerChild <i>number</i>
Default:	MaxThreadsPerChild 64
Context:	server config
Status:	MPM
Module:	perchild

This directive sets the maximum number of threads that will be created in each child process. To increase this value beyond its default, it is necessary to change the value of the [ThreadLimit](#) directive and stop and re-start the server.

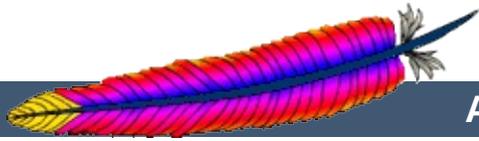


Description:	Total number of children alive at the same time
Syntax:	NumServers <i>number</i>
Default:	NumServers 2
Context:	server config
Status:	MPM
Module:	perchild

The **NumServers** directive determines the number of children alive at the same time. This number should be large enough to handle the requests for the entire site. To increase this value beyond the value of 8, it is necessary to change the value of the **ServerLimit** directive and stop and re-start the server. See the section above for a [configuration example](#).

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM prefork

Description:	Implements a non-threaded, pre-forking web server
Status:	MPM
Module Identifier:	mpm_prefork_module
Source File:	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.



Description:	Maximum number of idle child server processes
Syntax:	MaxSpareServers <i>number</i>
Default:	MaxSpareServers 10
Context:	server config
Status:	MPM
Module:	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 lower than `MinSpareServers`, Apache will automatically adjust it to `MinSpareServers + 1`.

See also

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



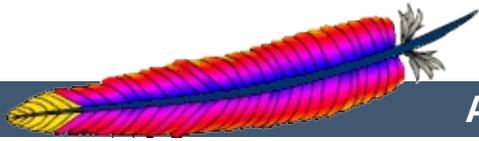
Description:	Minimum number of idle child server processes
Syntax:	<code>MinSpareServers</code> <i>number</i>
Default:	<code>MinSpareServers</code> 5
Context:	server config
Status:	MPM
Module:	prefork

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM threadpool

Description:	Yet another experimental variant of the standard worker MPM
Status:	MPM
Module Identifier:	mpm_threadpool_module
Source File:	threadpool.c

Summary

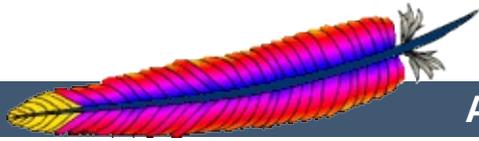
Warning

This MPM is a developer playground and highly experimental, so it may or may not work as expected.

This is an experimental variant of the standard [worker](#) MPM. Rather than queuing connections like the [worker](#) MPM, the [threadpool](#) MPM queues idle worker threads and hands each accepted connection to the next available worker.

The [threadpool](#) MPM can't match the performance of the [worker](#) MPM in benchmark testing. As of 2.0.39, some of the key load-throttling concepts from the [threadpool](#) MPM have been incorporated into the [worker](#) MPM. The [threadpool](#) code is useful primarily as a research platform. For general-purpose use and for any production environments, use [worker](#) instead.

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM winnt

Description:	This Multi-Processing Module is optimized for Windows NT.
Status:	MPM
Module Identifier:	mpm_winnt_module
Source File:	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

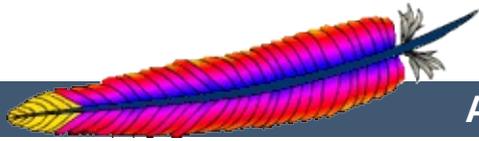


Description:	Use <code>accept()</code> rather than <code>AcceptEx()</code> to accept network connections
Syntax:	<code>Win32DisableAcceptEx</code>
Default:	<code>AcceptEx()</code> is enabled by default. Use this directive to disable use of <code>AcceptEx()</code>
Context:	server config
Status:	MPM
Module:	<code>mpm_winnt</code>
Compatibility:	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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache MPM worker

Description:	Multi-Processing Module implementing a hybrid multi-threaded multi-process web server
Status:	MPM
Module Identifier:	mpm_worker_module
Source File:	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 less system resources than a process-based server. Yet 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 launched is set by the [StartServers](#) directive. Then 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 by 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:

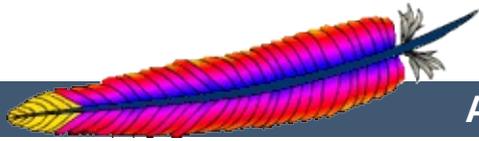
```
ServerLimit 16
StartServers 2
MaxClients 150
MinSpareThreads 25
MaxSpareThreads 75
ThreadsPerChild 25
```

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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_access`

Description:	Provides access control based on client hostname, IP address, or other characteristics of the client request.
Status:	Base
Module Identifier:	<code>access_module</code>
Source File:	<code>mod_access.c</code>
Compatibility:	Available only in versions prior to 2.1

Summary

The directives provided by `mod_access` are used in `<Directory>`, `<Files>`, and `<Location>` sections as well as `.htaccess` files to control access to particular parts of the server. Access can be controlled based on the client hostname, IP address, or other characteristics of the client request, as captured in `environment variables`. The `Allow` and `Deny` directives are used to specify which clients are or are not allowed access to the server, while the `Order` directive sets the default access state, and configures how the `Allow` and `Deny` directives interact with each other.

Both host-based access restrictions and password-based authentication may be implemented simultaneously. In that case, the `Satisfy` directive is used to determine how the two sets of restrictions interact.

In general, access restriction directives apply to all access methods (GET, PUT, POST, etc). This is the desired behavior in most cases. However, it is possible to restrict some methods, while leaving other methods unrestricted, by enclosing the directives in a `<Limit>` section.

See also

[Satisfy](#)

[Require](#)



Description:	Controls which hosts can access an area of the server
Syntax:	Allow from all <i>host</i> env= <i>env-variable</i> [<i>host</i> env= <i>env-variable</i>] ...
Context:	directory, .htaccess
Override:	Limit
Status:	Base
Module:	mod_access

The **Allow** directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, IP address range, or by other characteristics of the client request captured in environment variables.

The first argument to this directive is always `from`. The subsequent arguments can take three different forms. If `Allow from all` is specified, then all hosts are allowed access, subject to the configuration of the **Deny** and **Order** directives as discussed below. To allow only particular hosts or groups of hosts to access the server, the *host* can be specified in any of the following formats:

A (partial) domain-name

Example:

```
Allow from apache.org
Allow from .net example.edu
```

Hosts whose names match, or end in, this string are allowed access. Only complete components are matched, so the above example will match `foo.apache.org` but it will not match `fooapache.org`. This configuration will cause

Apache to perform a double reverse DNS lookup on the client IP address, regardless of the setting of the [HostnameLookups](#) directive. It will do a reverse DNS lookup on the IP address to find the associated hostname, and then do a forward lookup on the hostname to assure that it matches the original IP address. Only if the forward and reverse DNS are consistent and the hostname matches will access be allowed.

A full IP address

Example:

```
Allow from 10.1.2.3  
Allow from 192.168.1.104 192.168.1.205
```

An IP address of a host allowed access

A partial IP address

Example:

```
Allow from 10.1  
Allow from 10 172.20 192.168.2
```

The first 1 to 3 bytes of an IP address, for subnet restriction.

A network/netmask pair

Example:

```
Allow from 10.1.0.0/255.255.0.0
```

A network a.b.c.d, and a netmask w.x.y.z. For more fine-grained subnet restriction.

A network/nnn CIDR specification

Example:

```
Allow from 10.1.0.0/16
```

Similar to the previous case, except the netmask consists of *nnn* high-order 1 bits.

Note that the last three examples above match exactly the same set of hosts.

IPv6 addresses and IPv6 subnets can be specified as shown below:

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

The third format of the arguments to the **Allow** directive allows access to the server to be controlled based on the existence of an [environment variable](#). When `Allow from env=env-variable` is specified, then the request is allowed access if the environment variable *env-variable* exists. The server provides the ability to set environment variables in a flexible way based on characteristics of the client request using the directives provided by [mod_setenvif](#). Therefore, this directive can be used to allow access based on such factors as the clients User-Agent (browser type), Referer, or other HTTP request header fields.

Example:

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

In this case, browsers with a user-agent string beginning with

KnockKnock/2.0 will be allowed access, and all others will be denied.



Deny Directive

Description: Controls which hosts are denied access to the server

Syntax: Deny from *all|host|env=env-variable*
[host|env=env-variable] ...

Context: directory, .htaccess

Override: Limit

Status: Base

Module: mod_access

This directive allows access to the server to be restricted based on hostname, IP address, or environment variables. The arguments for the **Deny** directive are identical to the arguments for the **Allow** directive.



Description: Controls the default access state and the order in which **Allow** and **Deny** are evaluated.

Syntax: Order *ordering*

Default: Order Deny, Allow

Context: directory, .htaccess

Override: Limit

Status: Base

Module: mod_access

The **Order** directive, along with the **Allow** and **Deny** directives, controls a three-pass access control system. The first pass processes either all **Allow** or all **Deny** directives, as specified by the **Order** directive. The second pass parses the rest of the directives (**Deny** or **Allow**). The third pass applies to all requests which do not match either of the first two.

Note that all **Allow** and **Deny** directives are processed, unlike a typical firewall, where only the first match is used. The last match is effective (also unlike a typical firewall). Additionally, the order in which lines appear in the configuration files is not significant -- all **Allow** lines are processed as one group, all **Deny** lines are considered as another, and the default state is considered by itself.

Ordering is one of:

Allow, Deny

First, all **Allow** directives are evaluated; at least one must match, or the request is rejected. Next, all **Deny** directives are evaluated. If any matches, the request is rejected. Last, any requests which do not match an **Allow** or a **Deny** directive are denied by default.

Deny, Allow

First, all [Deny](#) directives are evaluated; if any match, the request is denied **unless** it also matches an [Allow](#) directive. Any requests which do not match any [Allow](#) or [Deny](#) directives are permitted.

Mutual-failure

This order has the same effect as `Order Allow, Deny` and is deprecated in its favor.

Keywords may only be separated by a comma; *no whitespace* is allowed between them.

Match	Allow,Deny result	Deny,Allow result
Match Allow only	Request allowed	Request allowed
Match Deny only	Request denied	Request denied
No match	Default to second directive: Denied	Default to second directive: Allowed
Match both Allow & Deny	Final match controls: Denied	Final match controls: Allowed

In the following example, all hosts in the `apache.org` domain are allowed access; all other hosts are denied access.

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

In the next example, all hosts in the `apache.org` domain are allowed access, except for the hosts which are in the `foo.apache.org` subdomain, who are denied access. All hosts not in the `apache.org` domain are denied access because the default state is to [Deny](#) access to the server.

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

On the other hand, if the **Order** in the last example is changed to **Deny, Allow**, all hosts will be allowed access. This happens because, regardless of the actual ordering of the directives in the configuration file, the **Allow from apache.org** will be evaluated last and will override the **Deny from foo.apache.org**. All hosts not in the **apache.org** domain will also be allowed access because the default state is **Allow**.

The presence of an **Order** directive can affect access to a part of the server even in the absence of accompanying **Allow** and **Deny** directives because of its effect on the default access state. For example,

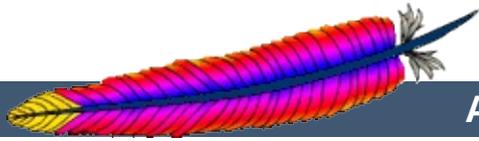
```
<Directory /www>
  Order Allow,Deny
</Directory>
```

will **Deny** all access to the **/www** directory because the default access state is set to **Deny**.

The **Order** directive controls the order of access directive processing only within each phase of the server's configuration processing. This implies, for example, that an **Allow** or **Deny** directive occurring in a **<Location>** section will always be evaluated after an **Allow** or **Deny** directive occurring in a **<Directory>** section or **.htaccess** file, regardless of the setting of the **Order** directive. For details on the merging of configuration sections, see the documentation on [How Directory, Location and Files sections work](#).

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

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_actions

- CGI .
- Base
- actions_module
- mod_actions.c

. Action CGI .
CGI . .

mod_cgi
CGI



```

: content-type CGI
: Action action-type cgi-script
: , , directory, .htaccess
Override : FileInfo
: Base
: mod_actions

```

```

    action-type cgi-script .
ScriptAlias AddHandler CGI URL.
type MIME content type . PATH_INFO
PATH_TRANSLATED CGI URL .

```

```

# :
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
.

```

- [AddHandler](#)



Script

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

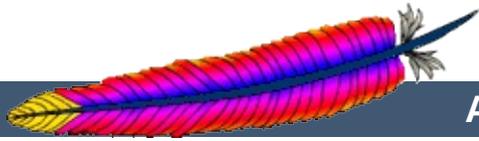
```
ScriptAlias AddHandler CGI URL.
Script PATH_INFO PATH_TRANSLATED CGI URL
```

```
Script .
Script put .
```

```
Script .
Script GET Script ( CGI , , foo.htm
```

```
# <ISINDEX>
Script GET /cgi-bin/search

# CGI PUT
Script PUT /~bob/put.cgi
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_alias

- , URL
- Base
- alias_module
- mod_alias.c

URL .

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

:	URL
:	RedirectMatch [<i>status</i>] <i>regex</i> URL
:	, , directory, .htaccess
Override :	FileInfo
:	Base
:	mod_alias

[Redirect](#) , URL .
URL , . ,
JPEG :

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```



Redirect

```
URL  
RedirectPermanent URL-path URL  
, , directory, .htaccess  
Override : FileInfo  
Base  
mod_alias
```

(301) . Redirect



RedirectTemp

```
: URL
: RedirectTemp URL-path URL
: , , directory, .htaccess
Override : FileInfo
: Base
: mod_alias
```

(302) . Redirect



ScriptAlias

```
: URL CGI
: ScriptAlias URL-path file-path|directory-path
: ,
: Base
: mod_alias
```

```
ScriptAlias Alias , mod
script CGI . URL-path (%) URL
.
```

```
:
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

```
http://myserver/cgi-bin/foo /web/cgi-
bin/foo .
```

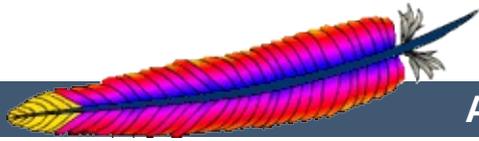


ScriptAliasMatch

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

```
ScriptAlias URL .
URL , . ,
/cgi-bin :
```

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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 .
```

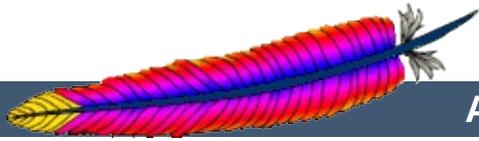
```
AddHandler send-as-is asis
```

```
.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 . .
```



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_auth

Description:	User authentication using text files
Status:	Base
Module Identifier:	auth_module
Source File:	mod_auth.c
Compatibility:	Available only in versions prior to 2.1

Summary

This module allows the use of HTTP Basic Authentication to restrict access by looking up users in plain text password and group files. Similar functionality and greater scalability is provided by [mod_auth_dbm](#). HTTP Digest Authentication is provided by [mod_auth_digest](#).

See also

[Require](#)

[Satisfy](#)

[AuthName](#)

[AuthType](#)



Description:	Sets whether authorization and authentication are passed to lower level modules
Syntax:	AuthAuthoritative On Off
Default:	AuthAuthoritative On
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Base
Module:	mod_auth

Setting the `AuthAuthoritative` directive explicitly to `Off` allows for both authentication and authorization to be passed on to lower level modules (as defined in the `modules.c` files) if there is **no** `userID` or `rule` matching the supplied `userID`. If there is a `userID` and/or `rule` specified; the usual password and access checks will be applied and a failure will give an "Authentication Required" reply.

So if a `userID` appears in the database of more than one module; or if a valid `Require` directive applies to more than one module; then the first module will verify the credentials; and no access is passed on; regardless of the `AuthAuthoritative` setting.

A common use for this is in conjunction with one of the database modules; such as `mod_auth_dbm`, `mod_auth_mysql`, and `mod_auth_anon`. These modules supply the bulk of the user credential checking; but a few (administrator) related accesses fall through to a lower level with a well protected `AuthUserFile`.

By default control is not passed on and an unknown `userID` or `rule` will result in an "Authentication Required" reply. Not setting it thus keeps the system secure and forces an NCSA compliant behaviour.

Security

Do consider the implications of allowing a user to allow fall-through in his .htaccess file; and verify that this is really what you want; Generally it is easier to just secure a single .htpasswd file, than it is to secure a database such as mSQL. Make sure that the [AuthUserFile](#) and the [AuthGroupFile](#) are stored outside the document tree of the web-server; do *not* put them in the directory that they protect. Otherwise, clients will be able to download the [AuthUserFile](#) and the [AuthGroupFile](#).



Description:	Sets the name of a text file containing the list of user groups for authentication
Syntax:	<code>AuthGroupFile file-path</code>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Base
Module:	mod_auth

The `AuthGroupFile` directive sets the name of a textual file containing the list of user groups for user authentication. *File-path* is the path to the group file. If it is not absolute, it is treated as relative to the `ServerRoot`.

Each line of the group file contains a groupname followed by a colon, followed by the member usernames separated by spaces.

Example:

```
mygroup: bob joe anne
```

Note that searching large text files is *very* inefficient; `AuthDBMGroupFile` provides a much better performance.

Security

Make sure that the `AuthGroupFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients may be able to download the `AuthGroupFile`.



Description:	Sets the name of a text file containing the list of users and passwords for authentication
Syntax:	<code>AuthUserFile <i>file-path</i></code>
Context:	directory, <code>.htaccess</code>
Override:	<code>AuthConfig</code>
Status:	Base
Module:	<code>mod_auth</code>

The `AuthUserFile` directive sets the name of a textual file containing the list of users and passwords for user authentication. *File-path* is the path to the user file. If it is not absolute (*i.e.*, if it doesn't begin with a slash), it is treated as relative to the `ServerRoot`.

Each line of the user file contains a username followed by a colon, followed by the encrypted password. If the same user ID is defined multiple times, `mod_auth` will use the first occurrence to verify the password.

The utility `htpasswd` which is installed as part of the binary distribution, or which can be found in `src/support`, is used to maintain this password file. See the [man page](#) for more details. In short:

Create a password file `Filename` with `username` as the initial ID. It will prompt for the password:

```
htpasswd -c Filename username
```

Add or modify `username2` in the password file `Filename`:

```
htpasswd Filename username2
```

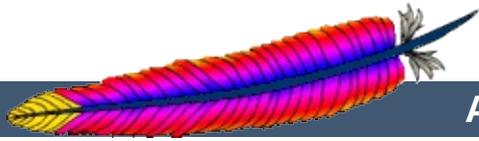
Note that searching large text files is *very* inefficient; [AuthDBMUserFile](#) should be used instead.

Security

Make sure that the [AuthUserFile](#) is stored outside the document tree of the web-server. Do **not** put it in the directory that it protects. Otherwise, clients may be able to download the [AuthUserFile](#).

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_auth_anon

Description:	Allows "anonymous" user access to authenticated areas
Status:	Extension
Module Identifier:	auth_anon_module
Source File:	mod_auth_anon.c
Compatibility:	Available only in versions prior to 2.1

Summary

This module does access control in a manner similar to anonymous-ftp sites; *i.e.* have a 'magic' user id 'anonymous' and the email address as a password. These email addresses can be logged.

Combined with other (database) access control methods, this allows for effective user tracking and customization according to a user profile while still keeping the site open for 'unregistered' users. One advantage of using Auth-based user tracking is that, unlike magic-cookies and funny URL pre/postfixes, it is completely browser independent and it allows users to share URLs.



The example below (when combined with the Auth directives of a htpasswd-file based (or GDM, mSQL etc.) base access control system allows users in as 'guests' with the following properties:

- It insists that the user enters a userID.
(Anonymous_NoUserID)
- It insists that the user enters a password.
(Anonymous_MustGiveEmail)
- The password entered must be a valid email address, ie. contain at least one '@' and a '.'.
(Anonymous_VerifyEmail)
- The userID must be one of anonymous guest www test welcome and comparison is **not** case sensitive.
(Anonymous)
- And the Email addresses entered in the passwd field are logged to the error log file. (Anonymous_LogEmail)

Excerpt of httpd.conf:

```
Anonymous_NoUserID off
Anonymous_MustGiveEmail on
Anonymous_VerifyEmail on
Anonymous_LogEmail on
Anonymous anonymous guest www test welcome

AuthName "Use 'anonymous' & Email address for guest entry"
AuthType basic

# An AuthUserFile/AuthDBUserFile/AuthDBMUserFile
# directive must be specified, or use
# Anonymous_Authoritative for public access.
# In the .htaccess for the public directory, add:
<Files *>
    Order Deny,Allow
    Allow from all

    Require valid-user
</Files>
```



Description:	Specifies userIDs that are allowed access without password verification
Syntax:	Anonymous <i>user</i> [<i>user</i>] ...
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_anon

A list of one or more 'magic' userIDs which are allowed access without password verification. The userIDs are space separated. It is possible to use the ' and " quotes to allow a space in a userID as well as the \ escape character.

Please note that the comparison is **case-IN-sensitive**.

I strongly suggest that the magic username 'anonymous' is always one of the allowed userIDs.

Example:

```
Anonymous anonymous "Not Registered" "I don't know"
```

This would allow the user to enter without password verification by using the userIDs "anonymous", "AnonyMous", "Not Registered" and "I Don't Know".



AuthConfig: Anonymous_Authoritative Directive

Description:	Configures if authorization will fall-through to other methods
Syntax:	Anonymous_Authoritative On Off
Default:	Anonymous_Authoritative Off
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_anon

When set On, there is no fall-through to other authentication methods. So if a userID does not match the values specified in the [Anonymous](#) directive, access is denied.

Be sure you know what you are doing when you decide to switch it on. And remember that the order in which the Authentication modules are queried is defined in the modules.c files at compile time.



Anonymous_LogEmail

Description:	Sets whether the password entered will be logged in the error log
Syntax:	Anonymous_LogEmail On Off
Default:	Anonymous_LogEmail On
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_anon

When set On, the default, the 'password' entered (which hopefully contains a sensible email address) is logged in the error log.



Description:	Specifies whether blank passwords are allowed
Syntax:	Anonymous_MustGiveEmail On Off
Default:	Anonymous_MustGiveEmail On
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_anon

Specifies whether the user must specify an email address as the password. This prohibits blank passwords.



Description:	Sets whether the userID field may be empty
Syntax:	Anonymous_NoUserID On Off
Default:	Anonymous_NoUserID Off
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_anon

When set On, users can leave the userID (and perhaps the password field) empty. This can be very convenient for MS-Explorer users who can just hit return or click directly on the OK button; which seems a natural reaction.

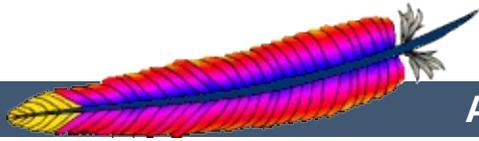


Description:	Sets whether to check the password field for a correctly formatted email address
Syntax:	Anonymous_VerifyEmail On Off
Default:	Anonymous_VerifyEmail Off
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_anon

When set On the 'password' entered is checked for at least one '@' and a '.' to encourage users to enter valid email addresses (see the above [Anonymous_LogEmail](#)).

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_auth_dbm

Description:	Provides for user authentication using DBM files
Status:	Extension
Module Identifier:	auth_dbm_module
Source File:	mod_auth_dbm.c
Compatibility:	Available only in versions prior to 2.1

Summary

This module provides for HTTP Basic Authentication, where the usernames and passwords are stored in DBM type database files. It is an alternative to the plain text password files provided by [mod_auth](#).

See also

[AuthName](#)

[AuthType](#)

[Require](#)

[Satisfy](#)



Description:	Sets whether authentication and authorization will be passed on to lower level modules
Syntax:	AuthDBMAuthoritative On Off
Default:	AuthDBMAuthoritative On
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_dbm

Setting the `AuthDBMAuthoritative` directive explicitly to `Off` allows for both authentication and authorization to be passed on to lower level modules (as defined in the `modules.c` files) if there is **no** `userID` or `rule` matching the supplied `userID`. If there is a `userID` and/or `rule` specified; the usual password and access checks will be applied and a failure will give an "Authentication Required" reply.

So if a `userID` appears in the database of more than one module; or if a valid `Require` directive applies to more than one module; then the first module will verify the credentials; and no access is passed on; regardless of the `AuthDBMAuthoritative` setting.

A common use for this is in conjunction with one of the basic auth modules; such as `mod_auth`. Whereas this DBM module supplies the bulk of the user credential checking; a few (administrator) related accesses fall through to a lower level with a well protected `.htpasswd` file.

By default, control is not passed on and an unknown `userID` or `rule` will result in an "Authentication Required" reply. Not setting it thus keeps the system secure and forces an NCSA compliant behaviour.

Security:

Do consider the implications of allowing a user to allow fall-through in his `.htaccess` file; and verify that this is really what you want; Generally it is easier to just secure a single `.htpasswd` file, than it is to secure a database which might have more access interfaces.



Description:	Sets the name of the database file containing the list of user groups for authentication
Syntax:	<code>AuthDBMGroupFile <i>file-path</i></code>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_dbm

The `AuthDBMGroupFile` directive sets the name of a DBM file containing the list of user groups for user authentication. *File-path* is the absolute path to the group file.

The group file is keyed on the username. The value for a user is a comma-separated list of the groups to which the users belongs. There must be no whitespace within the value, and it must never contain any colons.

Security: make sure that the `AuthDBMGroupFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients will be able to download the `AuthDBMGroupFile` unless otherwise protected.

Combining Group and Password DBM files: In some cases it is easier to manage a single database which contains both the password and group details for each user. This simplifies any support programs that need to be written: they now only have to deal with writing to and locking a single DBM file. This can be accomplished by first setting the group and password files to point to the same DBM:

```
AuthDBMGroupFile /www/userbase
AuthDBMUserFile /www/userbase
```

The key for the single DBM is the username. The value consists of

```
Unix Crypt-ed Password:List of Groups[:(ignored)]
```

The password section contains the encrypted password as before. This is followed by a colon and the comma separated list of groups. Other data may optionally be left in the DBM file after another colon; it is ignored by the authentication module. This is what www.telescope.org uses for its combined password and group database.



AuthDBMType Directive

Description:	Sets the type of database file that is used to store passwords
Syntax:	AuthDBMType default SDBM GDBM NDBM DB
Default:	AuthDBMType default
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_dbm
Compatibility:	Available in version 2.0.30 and later.

Sets the type of database file that is used to store the passwords. The default database type is determined at compile time. The availability of other types of database files also depends on [compile-time settings](#).

It is crucial that whatever program you use to create your password files is configured to use the same type of database.



Description:	Sets the name of a database file containing the list of users and passwords for authentication
Syntax:	<code>AuthDBMUserFile <i>file-path</i></code>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_auth_dbm

The `AuthDBMUserFile` directive sets the name of a DBM file containing the list of users and passwords for user authentication. *File-path* is the absolute path to the user file.

The user file is keyed on the username. The value for a user is the encrypted password, optionally followed by a colon and arbitrary data. The colon and the data following it will be ignored by the server.

Security:

Make sure that the `AuthDBMUserFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients will be able to download the `AuthDBMUserFile`.

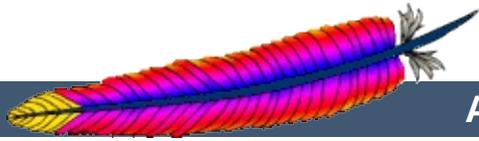
Important compatibility note: The implementation of "dbmopen" in the apache modules reads the string length of the hashed values from the DBM data structures, rather than relying upon the string being NULL-appended. Some applications, such as the Netscape web server, rely upon the string being NULL-appended, so if you are having trouble using DBM files interchangeably between applications this may be a part of the problem.

A perl script called `dbmmanage` is included with Apache. This

program can be used to create and update DBM format password files for use with this module.

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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 .
[AuthGroupFile](#) [AuthDigestGroupFile](#) , AuthType Basic
[AuthUserFile](#) AuthType Digest [AuthDigestFile](#)
URI [AuthDi](#)

[htdigest](#) () .

```
:  
<Location /private/>  
  AuthType Digest  
  AuthName "private area"  
  AuthDigestDomain /private/ http://mirror.my.dom/private2/  
  AuthDigestFile /web/auth/.digest_pw  
  Require valid-user  
</Location>
```

Digest authentication Basic authentication ,
2002 11 digest authentication
- " [MS Internet Explorer](#) ") [M](#)
[Internet Explorer](#), [Amaya](#), [Mozilla](#), 7 [Netscape](#) .
digest authentication basic authentication
.



Internet Explorer Digest authentication

Internet Explorer Digest authentication RFC G

GET POST .

, 2.0.51 AuthDigestEnableQueryStringHack
. AuthDigestEnableQueryStringHack M
URI digest . . .

MSIE Digest Authentication :
BrowserMatch "MSIE" AuthDigestEnableQueryStringHack=On

BrowserMatch .



AuthDigestAlgorithm

:	digest authentication challenge response hash
:	AuthDigestAlgorithm MD5 MD5-sess
:	AuthDigestAlgorithm MD5
:	directory, .htaccess
Override :	AuthConfig
:	Experimental
:	mod_auth_digest

AuthDigestAlgorithm challenge response hash

.

MD5-sess .



AuthDigestDomain

:	digest authentication URI
:	AuthDigestDomain <i>URI</i> [<i>URI</i>] ...
:	directory, .htaccess
Override :	AuthConfig
:	Experimental
:	mod_auth_digest

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



AuthDigestFile

:	digest authentication
:	AuthDigestFile <i>file-path</i>
:	directory, .htaccess
Override :	AuthConfig
:	Experimental
:	mod_auth_digest

```
AuthDigestFile digest authentication
.      File-path      .
.
.      support/      htdigest
```



AuthDigestGroupFile

```
: digest authentication
: AuthDigestGroupFile file-path
: directory, .htaccess
Override : AuthConfig
: Experimental
: mod_auth_digest
```

```
AuthDigestGroupFile          () .
path .
```

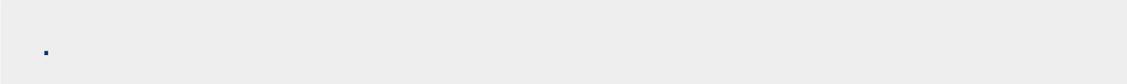
```
, . .
mygroup: bob joe anne
```

```
:
AuthGroupFile .
AuthGroupFile .
```



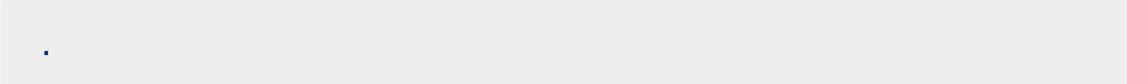
AuthDigestNcCheck

```
: 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 .
```



AuthDigestQop

:	digest authentication (quality-of-protection)
:	.
:	AuthDigestQop none auth auth-int [auth auth-int]
:	AuthDigestQop auth
:	directory, .htaccess
Override :	AuthConfig
:	Experimental
:	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

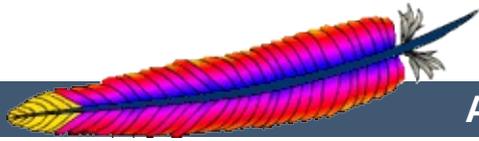
AuthDigestShmemSize 0

size , K M KBytes MBytes .
, :

```

AuthDigestShmemSize 1048576
AuthDigestShmemSize 1024K
AuthDigestShmemSize 1M

```



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_auth_ldap`

Description:	Allows an LDAP directory to be used to store the database for HTTP Basic authentication.
Status:	Experimental
Module Identifier:	<code>auth_ldap_module</code>
Source File:	<code>mod_auth_ldap.c</code>
Compatibility:	Available in version 2.0.41 and later

Summary

`mod_auth_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.
- Support for Microsoft FrontPage allows FrontPage users to control access to their webs, while retaining LDAP for user authentication.
- 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).

See also

[mod_ldap](#)



-
- [Operation](#)
 - [The Authentication Phase](#)
 - [The Authorization Phase](#)
 - [The Require Directives](#)
 - [Require valid-user](#)
 - [Require user](#)
 - [Require group](#)
 - [Require dn](#)
 - [Require ldap-attribute](#)
 - [Examples](#)
 - [Using TLS](#)
 - [Using SSL](#)
 - [Using Microsoft FrontPage with mod_auth_ldap](#)
 - [How It Works](#)
 - [Caveats](#)



There are two phases in granting access to a user. The first phase is authentication, in which [mod_auth_ldap](#) verifies that the user's credentials are valid. This is also called the *search/bind* phase. The second phase is authorization, in which [mod_auth_ldap](#) determines if the authenticated user is allowed access to the resource in question. This is also known as the *compare* phase.

The Authentication Phase

During the authentication phase, [mod_auth_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_auth_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

[AuthLDAPURL](#)

Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to

	use.
AuthLDAPBindDN	An optional DN to bind with during the search phase.
AuthLDAPBindPassword	An optional password to bind with during the search phase.

The Authorization Phase

During the authorization phase, [mod_auth_ldap](#) attempts to determine if the user is authorized to access the resource. Many of these checks require [mod_auth_ldap](#) to do a compare operation on the LDAP server. This is why this phase is often referred to as the compare phase. [mod_auth_ldap](#) accepts the following [Require](#) directives to determine if the credentials are acceptable:

- Grant access if there is a [Require valid-user](#) directive.
- Grant access if there is a [Require user](#) directive, and the username in the directive matches the username passed by the client.
- Grant access if there is a [Require dn](#) directive, and the DN in the directive matches the DN fetched from the LDAP directory.
- Grant access if there is a [Require 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.
- otherwise, deny or decline access

[mod_auth_ldap](#) uses the following directives during the compare phase:

AuthLDAPURL	The attribute specified in the
-----------------------------	--------------------------------

URL is used in compare operations for the Require user operation.

AuthLDAPCompareDNOnServer

Determines the behavior of the Require dn directive.

AuthLDAPGroupAttribute

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

AuthLDAPGroupAttributeIsDN

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



Apache's [Require](#) directives are used during the authorization phase to ensure that a user is allowed to access a resource.

Require valid-user

If this directive exists, [mod_auth_ldap](#) grants access to any user that has successfully authenticated during the search/bind phase.

Require user

The `Require user` directive specifies what usernames can access the resource. Once [mod_auth_ldap](#) has retrieved a unique DN from the directory, it does an LDAP compare operation using the username specified in the `Require 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 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 user "Barbara Jenson"  
Require user "Fred User"  
Require user "Joe Manager"
```

Because of the way that [mod_auth_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 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 user bjenson fuser jmanager
```

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

Behavior of this directive is modified by the [AuthLDAPGroupAttribute](#) and [AuthLDAPGroupAttributeIsDN](#) directives.

Require dn

The `Require 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 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 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 a 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
```



- 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 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 filter, 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. The text in blue is the part that is filled in using the attribute specified in the URL. The text in red is the part that is filled in using the filter specified in the URL. The text in green is filled in using the information that is retrieved from the HTTP client. 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 [LDAPTrustedCA](#) and [LDAPTrustedCAType](#).



To use SSL, see the [mod_ldap](#) directives [LDAPTrustedCA](#) and [LDAPTrustedCAType](#).

To specify a secure LDAP server, use *ldaps://* in the [AuthLDAPURL](#) directive, instead of *ldap://*.



Normally, FrontPage uses FrontPage-web-specific user/group files (i.e., the `mod_auth` module) 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"  
AuthLDAPAuthoritative off  
AuthLDAPFrontPageHack on
```

`AuthLDAPAuthoritative` must be off to allow `mod_auth_ldap` to decline group authentication so that Apache will fall back to file authentication for checking group membership. This allows the FrontPage-managed group file to be used.

How It Works

FrontPage restricts access to a web by adding the `Require valid-user` directive to the `.htaccess` files. If `AuthLDAPFrontPageHack` is not on, 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. The purpose of the hack is to force Apache to consult the local user file (which is managed by FrontPage) - instead of LDAP - when handling the `Require valid-user` directive.

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_auth` 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` in order to use FrontPage support. This is because Apache will still use the `mod_auth` 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_auth_ldap` has to be able to grab the `AuthUserFile` directive that is found in FrontPage `.htaccess` files so that it knows where to look for the valid user list. If the `mod_auth_ldap` directives aren't in the same `.htaccess` file as the FrontPage directives, then the hack won't work, because `mod_auth_ldap` will never get a chance to process the `.htaccess` file, and won't be able to find the FrontPage-managed user file.



Description:	Prevent other authentication modules from authenticating the user if this one fails
Syntax:	AuthLDAPAuthoritative on off
Default:	AuthLDAPAuthoritative on
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

Set to off if this module should let other authentication modules attempt to authenticate the user, should authentication 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).



Description:	Optional DN to use in binding to the LDAP server
Syntax:	AuthLDAPBindDN <i>distinguished-name</i>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

An optional DN used to bind to the server when searching for entries. If not provided, [mod_auth_ldap](#) will use an anonymous bind.



Description:	Password used in conjunction with the bind DN
Syntax:	AuthLDAPBindPassword <i>password</i>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

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.



AuthLDAPCharsetConfig Directive

Description:	Language to charset conversion configuration file
Syntax:	<code>AuthLDAPCharsetConfig <i>file-path</i></code>
Context:	server config
Status:	Experimental
Module:	<code>mod_auth_ldap</code>

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.



Description:	Use the LDAP server to compare the DN's
Syntax:	AuthLDAPCompareDNOnServer on off
Default:	AuthLDAPCompareDNOnServer on
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

When set, [mod_auth_ldap](#) will use the LDAP server to compare the DN's. This is the only foolproof way to compare DN's. [mod_auth_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_auth_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.



Description:	When will the module de-reference aliases
Syntax:	AuthLDAPDereferenceAliases never searching finding always
Default:	AuthLDAPDereferenceAliases Always
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

This directive specifies when `mod_auth_ldap` will de-reference aliases during LDAP operations. The default is `always`.



Description:	Turn on or off LDAP authentication
Syntax:	AuthLDAPEnabled on off
Default:	AuthLDAPEnabled on
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

Set to off to disable [mod_auth_ldap](#) in certain directories. This is useful if you have [mod_auth_ldap](#) enabled at or near the top of your tree, but want to disable it completely in certain locations.



Description:	Allow LDAP authentication to work with MS FrontPage
Syntax:	AuthLDAPFrontPageHack on off
Default:	AuthLDAPFrontPageHack off
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_ldap

See the section on [using Microsoft FrontPage](#) with [mod_auth_ldap](#).



AuthLDAPGroupAttribute Directive

Description:	LDAP attributes used to check for group membership
Syntax:	AuthLDAPGroupAttribute <i>attribute</i>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_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_auth_ldap](#) uses the member and uniquemember attributes.



Description:	Use the DN of the client username when checking for group membership
Syntax:	AuthLDAPGroupAttributeIsDN on off
Default:	AuthLDAPGroupAttributeIsDN on
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_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_auth_ldap](#) will check if the group has cn=Babs Jenson, o=Airius as a member. If this directive is not set, then [mod_auth_ldap](#) will check if the group has bjenson as a member.



Description:	Use the DN of the client username to set the REMOTE_USER environment variable
Syntax:	AuthLDAPRemoteUserIsDN on off
Default:	AuthLDAPRemoteUserIsDN off
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_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.



Description:	URL specifying the LDAP search parameters
Syntax:	AuthLDAPUrl <i>url</i>
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Experimental
Module:	mod_auth_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_auth_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_auth_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 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))`.

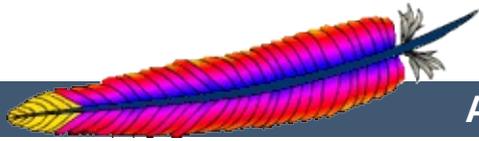
For example, consider an URL of `ldap://ldap.airius.com/o=Airius?cn?sub?(posixid=*)`. When a client attempts to connect using a

username of Babs Jenson, the resulting search filter will be (&
(posixid=*)(cn=Babs Jenson)).

See above for examples of [AuthLDAPURL](#) URLs.

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

```
:\n\nAddAlt string file [file] ...  
,\n,\ndirectory, .htaccess  
Override : Indexes  
:\nBase  
:\nmod_autoindex
```

```
AddAlt FancyIndexing . F  
, , , . String  
.\n,\n,\n.
```

```
AddAlt "PDF file" *.pdf  
AddAlt Compressed *.gz *.zip *.Z
```



Adding Encoding

```

: MIME-encoding
: AddAltByEncoding string MIME-encoding
  [MIME-encoding] ...
: , , directory, .htaccess
Override : Indexes
: Base
: mod_autoindex

```

```

AddAltByEncoding FancyIndexing .
MIME-encoding x-compress content-encoding. Str
( " ' ) . , ,
.

```

```

AddAltByEncoding gzip x-gzip

```



AddAltByType

```
:_ MIME content-type
:_ AddAltByType string MIME-type [MIME-
  type] ...
:_ , , directory, .htaccess
Override : Indexes
:_ Base
:_ mod_autoindex
```

```
AddAltByType FancyIndexing .
type text/html content-type. String ( "
') . , ,
.
```

```
AddAltByType 'plain text' text/plain
```



AddDescription

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

```
FancyIndexing . File ,  
String ( " ) .
```

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

```
23. IndexOptions SuppressIcon  
6 , IndexOptions SuppressSize 7 ,  
IndexOptions SuppressLastModified 19 .  
55 .
```

```
DescriptionWidth Inde
```

```
AddDescription character entity ( ; &lt; , & ;  
) HTML . ( )  
) .
```



ADDICON

```
:  
: AddIcon icon name [name] ...  
: , , directory, .htaccess  
Override : Indexes  
: Base  
: mod_autoindex
```

[FancyIndexing](#) *name* .
escaped) URL (*alttext, url*) . *alttext*

Name ^^DIRECTORY^^, ()
^^BLANKICON^^, , , .

```
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm  
AddIcon /icons/dir.xbm ^^DIRECTORY^^  
AddIcon /icons/backup.xbm *~
```

[AddIcon](#) [AddIconByType](#) .



Adding Encoding

```

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

```

```

FancyIndexing . Icon (%-esc:
URL (alttext,url) . alttext
.
MIME-encoding content-encoding .

```

```
AddIconByEncoding /icons/compress.xbm x-compress
```



AddIconByType

```
:_ MIME content-type
:_ AddIconByType icon MIME-type [MIME-
type] ...
:_ , , directory, .htaccess
Override : Indexes
:_ Base
:_ mod_autoindex
```

```
FancyIndexing MIME-type .
(%-escaped) URL (alttext,url) . alt
```

MIME-type mime type .

```
AddIconByType (IMG,/icons/image.xbm) image/*
```



DefaultIcon

```
DefaultIcon url-path
, , directory, .htaccess
Override : Indexes
Base
mod_autoindex
```

DefaultIcon [FancyIndexing](#) .
Icon (%-escaped) URL.

```
DefaultIcon /icon/unknown.xbm
```



HeaderName

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

HeaderName . Filename

```
HeaderName HEADER.html
```

```
HeaderName ReadmeName Filename
URI . Filename DocumentRoot
HeaderName /include/HEADER.html
Filename major content type text/* ( , text/html,
text/plain, ) . , (
text/html filename CGI :
AddType text/html .cgi
Options MultiViews . filename (CGI )
text/html options Includes IncludesNOEXEC
server-side includes . ( mod_include )
```

HeaderName (<html>, <head>,) HTML

[IndexOptions +SuppressHTMLPreamble](#)

.



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



ReadmeName

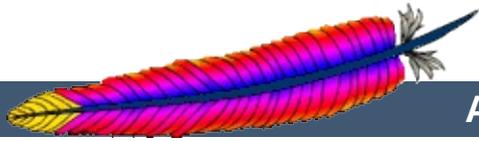
```
ReadmeName filename
, , directory,
.htaccess
Override : Indexes
Base
mod_autoindex
```

ReadmeName . Filename
 . Filename DocumentRoot .

```
ReadmeName FOOTER.html
```

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

HeaderName .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_cache

- Content cache keyed to URIs.
- 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_proxy mod_mem_cache , (

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:/cachroot
    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 /
```



CacheForceCompletion

```
CacheForceCompletion Percentage  
CacheForceCompletion 60  
,  
Experimental  
mod_cache
```

CacheForceCompletion

1 100 . 0 . 100
60 90 .

CacheForceCompletion 80

```
:
```



CacheIgnoreCacheControl

```
CacheIgnoreCacheControl On|Off  
CacheIgnoreCacheControl Off  
,  
Experimental  
mod_cache
```

no-cache no-store
CacheIgnoreCacheControl .
CacheIgnoreCacheControl On no-cache no-store

CacheIgnoreCacheControl On



CacheIgnoreHeaders

```
: Do not store the given HTTP header(s) in the cache.  
: CacheIgnoreHeaders header-string [header-string] ...  
: CacheIgnoreHeaders None  
:  
: Experimental  
: mod_cache
```

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



CacheIgnoreNoLastMod

```
[:] Last Modified .  
[:] CacheIgnoreNoLastMod On|Off  
[:] CacheIgnoreNoLastMod Off  
[:] ,  
[:] Experimental  
[:] mod_cache
```

```
. ( mo  
CacheIgnoreNoLastMod  
CacheDefaultExpire .
```

```
CacheIgnoreNoLastMod On
```



```
┆ 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
```



CacheMaxExpire

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

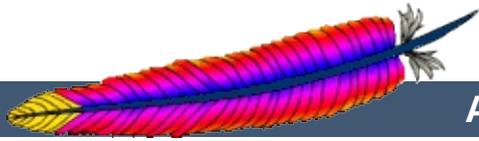
CacheMaxExpire

HTTP

, .

.

```
CacheMaxExpire 604800
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_cern_meta

- ┆ CERN
- ┆ Extension
- ┆ cern_meta_module
- ┆ mod_cern_meta.c

CERN .

, Expires:

CERN .

HTTP

[CERN metafile semantics](#) .

[mod_headers](#)

[mod_asis](#)



Module

:	CERN
:	MetaDir <i>directory</i>
:	MetaDir .web
:	, , directory, .htaccess
Override :	Indexes
:	Extension
:	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
Override :	Indexes
<input type="checkbox"/>	Extension
<input type="checkbox"/>	mod_cern_meta

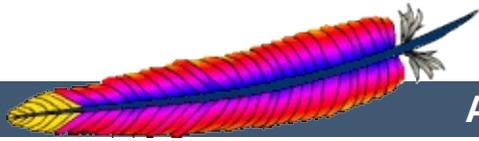
.



:	CERN
:	MetaSuffix <i>suffix</i>
:	MetaSuffix <i>.meta</i>
:	, , directory, .htaccess
Override :	Indexes
:	Extension
:	mod_cern_meta

. ,
 DOCUMENT_ROOT/somedir/index.html
 DOCUMENT_ROOT/somedir/.web/index.html.meta
 MIME .

:
 MetaSuffix *.meta*



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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 ,

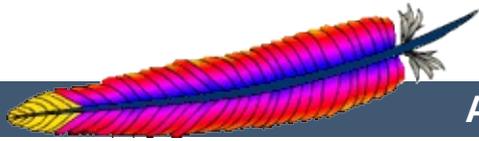
PUT POST



ScriptLogLength

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

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_cgid

- CGI
- CGI
- Base
- cgid_module
- mod_cgid.c
- MPMs

ScriptSock mod_cgid mod_cgi .
mod_cgi .

CGI (fork)
mod_cgid CGI .
(unix domain socket) .

MPM mod_cgi .
mod_cgi . cgi .

mod_cgi

ID CGI

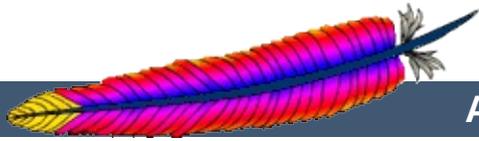


ScriptSock

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

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

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

```

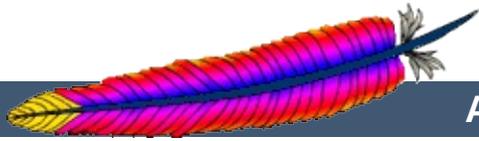
CharsetSourceEnc .
charset APR . iconv

```

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

```

Solaris 8 iconv .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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 .  
.
```



...
DavDepthInfinity

- [PROPFIND Depth: Infinity](#)
- [DavDepthInfinity on|off](#)
- [DavDepthInfinity off](#)
- [, , directory](#)
- [Extension](#)
- [mod_dav](#)

DavDepthInfinity

'Depth: Infinity'

PROPF

.

.



```

: DAV
: DavMinTimeout seconds
: DavMinTimeout 0
: , , directory
: Extension
: mod_dav

```

DAV (lock)

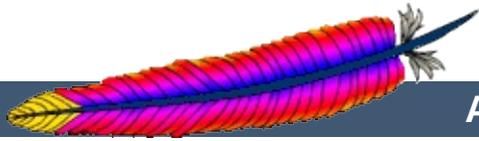
DavMinTimeout
Folders 120 .

() . Microsoft Web
DavMinTimeout (600)

```

<Location /MSWord>
  DavMinTimeout 600
</Location>

```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

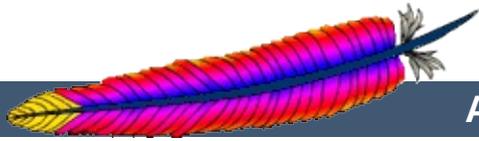
```

DavLockDB var/DavLock

```

User Group

DavLock



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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) .

`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 2.0.45

```

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



DeflateMemLevel

```
zlib
DeflateMemLevel value
DeflateMemLevel 9
,
Extension
mod_deflate
```

DeflateMemLevel zlib . (1 9

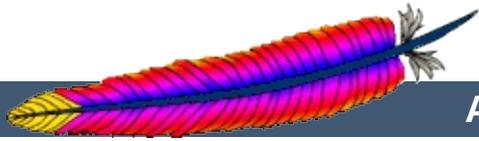


```

: Zlib window size
: DeflateWindowSize value
: DeflateWindowSize 15
: ,
: Extension
: mod_deflate

```

`DeflateWindowSize` zlib window size (1 15)
window size .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

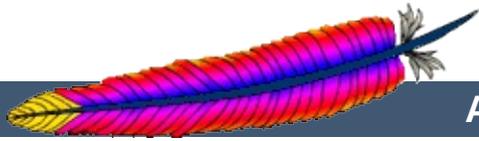
```
DirectorySlash On|Off
DirectorySlash On
, , directory, .htaccess
Override : Indexes
Base
mod_dir
2.0.51
```

```
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 .
```

| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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
```



CacheExpiryCheck

```
CacheExpiryCheck On|Off  
CacheExpiryCheck On  
,  
Experimental  
mod_disk_cache
```

CacheExpiryCheck Off

CacheExpiryCheck



```
URL
CacheGcClean hours url-string
CacheGcClean ?
,
Experimental
mod_disk_cache
```

CacheGcClean 12 /daily_scripts

CacheGcClean



CacheGcDaily

```
:(24
)
CacheGcDaily time
CacheGcDaily ?
,
Experimental
mod_disk_cache
```

.

CacheGcDaily 23:59

CacheGcDaily .



CacheGcInterval

```
CacheGcInterval .  
CacheGcInterval hours  
,  
Experimental  
mod_disk_cache
```

CacheGcInterval .

CacheGcInterval 24

CacheGcInterval .



CacheGcMemUsage

- CacheGcMemUsage (kilobyte)
- CacheGcMemUsage *KBytes*
- CacheGcMemUsage ?
- CacheGcMemUsage ,
- CacheGcMemUsage Experimental
- CacheGcMemUsage mod_disk_cache

CacheGcMemUsage 16

CacheGcMemUsage



```
URL .
CacheGcUnused hours url-string
CacheGcUnused ?
,
Experimental
mod_disk_cache
```

CacheGcUnused 12 /local_images

CacheGcUnused .



CACHEMAXFILESZ

```
:| ()  
:| CacheMaxFileSize bytes  
:| CacheMaxFileSize 1000000  
:| ,  
:| Experimental  
:| mod_disk_cache
```

CacheMaxFileSize .

```
CacheMaxFileSize 64000
```



```
⋮  ()  
⋮  CacheMinFileSize bytes  
⋮  CacheMinFileSize 1  
⋮  ,  
⋮  Experimental  
⋮  mod_disk_cache
```

CacheMinFileSize .

```
CacheMinFileSize 64
```



CACHEROOT

```
┆ root
┆ CacheRoot directory
┆ ,
┆ Experimental
┆ mod_disk_cache
```

```
CacheRoot . mod_d
CacheDirLevels CacheDirLength CacheRoot
root
```

```
CacheRoot c:/cacheroor
```



```
CacheSize (KByte
)
CacheSize KBytes
CacheSize 1000000
,
Experimental
mod_disk_cache
```

CacheSize KByte (1024) .

```
CacheSize 5000000
```

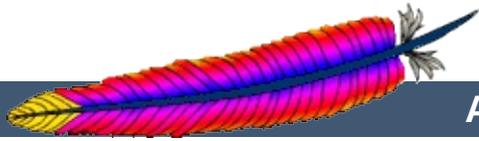


CacheTimeMargin

- CacheTimeMargin ?
- CacheTimeMargin ?
- ,
- Experimental
- mod_disk_cache

CacheTimeMargin X

CacheTimeMargin .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_dumpio

Description:	Dumps all I/O to error log as desired.
Status:	Experimental
Module Identifier:	dumpio_module
Source File:	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.

In order for dumping to work [LogLevel](#) must be set to debug.



DumpIOInput Directive

Description:	Dump all input data to the error log
Syntax:	DumpIOInput On Off
Default:	DumpIOInput Off
Context:	server config
Status:	Experimental
Module:	mod_dumpio
Compatibility:	DumpIOInput is only available in Apache 2.0.53 and later.

Enable dumping of all input.

```
Example  
DumpIOInput On
```



DumpIOOutput Directive

Description:	Dump all output data to the error log
Syntax:	DumpIOOutput On Off
Default:	DumpIOOutput Off
Context:	server config
Status:	Experimental
Module:	mod_dumpio
Compatibility:	DumpIOOutput is only available in Apache 2.0.53 and later.

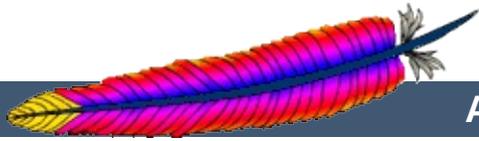
Enable dumping of all output.

Example

```
DumpIOOutput On
```

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_echo



- echo
- Experimental
- echo_module
- mod_echo.c
- Apache 2.0

.
, .

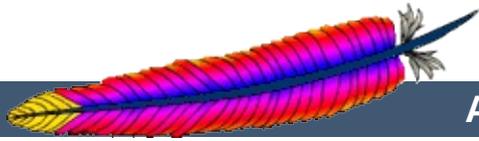
echo . telnet



```
ProtocolEcho
: echo
: ProtocolEcho On|Off
: ,
: Experimental
: mod_echo
: ProtocolEcho 2.0
.
```

ProtocolEcho echo .

```
ProtocolEcho On
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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
```

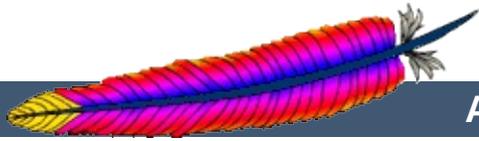


UNSETENV

```
:  
: UnsetEnv env-variable [env-variable]  
: ...  
: , , directory, .htaccess  
Override : FileInfo  
: Base  
: mod_env
```

CGI SSI .

```
UnsetEnv LD_LIBRARY_PATH
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_example

- API
- Experimental
- example_module
- mod_example.c

2.0

```
src/modules/example API
.
mod_example.c (callback)
. !
example . "example-handler"
example .
```



Example

example :

1. `src/Configuration` "AddModule
`modules/example/mod_example" . ,
.`
`AddModule modules/example/mod_example.o`
 2. `src/Configure` ("cd src; ./Configure")
`Makefile , src/modul
.`
 3. (`src "make").`
- :
- A. `mkdir src/modules/mymodule`
 - B. `cp src/modules/example/* src/modules/mymodule`
 - C. `.`
 - D. `[1] [3] .`



example srm.conf :

```
<Location /example-info>  
SetHandler example-handler  
</Location>
```

[.htaccess](#) , "test.example"

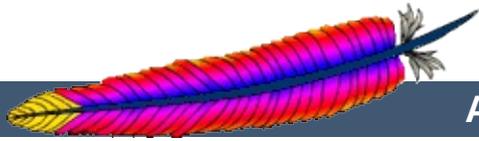
```
AddHandler example-handler .example
```



Example

```
API
Example
, , directory,
.htaccess
Experimental
mod_example
```

```
Example example
example URL
. "Example directive declared here:
YES/NO"
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

:	Expires
:	ExpiresActive On Off
:	, , directory, .htaccess
Override :	Indexes
:	Extension
:	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 .
( , ), .
```

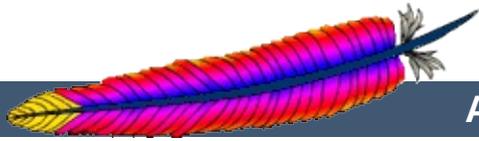
```
:
#
ExpiresActive On
# GIF
ExpiresByType image/gif A2592000
# HTML ExpiresByType text/html M604800
```

```
ExpiresActive On ExpiresDefa
MIME type .
```



ExpiresDefault

```
ExpiresDefault <code>seconds  
, , directory, .htaccess  
Override : Indexes  
Extension  
mod_expires
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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=input
.

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*

```

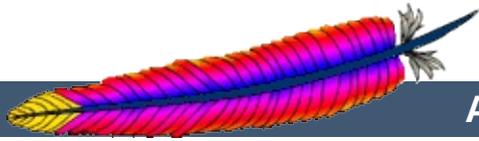
DebugLevel mod_ext_filter .
. DebugLevel=0 . ,
. mod_ext_filter.c DBGLV
.
: core LogLevel .
```

LogStderr | NoLogStderr

```

LogStderr .
NoLogStderr .
```

```
ExtFilterOptions LogStderr DebugLevel=0
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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()

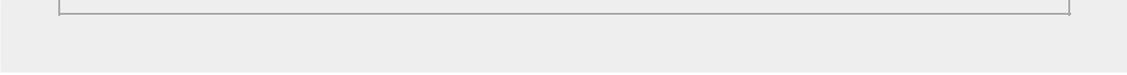
CacheFile

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 file-path [file-path] ...
:
: Experimental
: mod_file_cache

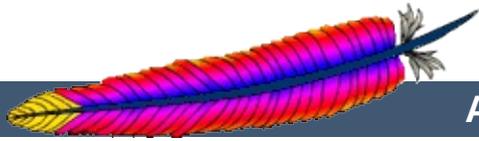
```

MMapFile () mmap()
 (unmap).
 .
file-path . URL-
 stat() inode .
mod_rewrite .

```

MMapFile /usr/local/apache/htdocs/index.html

```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_headers

- [: HTTP](#)
- [: Extension](#)
- [: headers_module](#)
- [: mod_headers.c](#)

HTTP

. , .



mod_headers

```
<Directory>, <Location>, <Files>, .htaccess
```

.

.

1.

2.

3. <Directory> .htaccess

4. <Files>

5. <Location>

.

.

```
RequestHeader append MirrorID "mirror 12"  
RequestHeader unset MirrorID
```

MirrorID

. MirrorID "mirror 12"

.



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] [env=[!]variable]
: , , directory, .htaccess
Override : FileInfo
: Extension
: mod_headers
: Condition 2.0.51

```

```

HTTP ,.
.
condition , onsuccess always .
. onsuccess 2xx , always ( 2x:
, .
.
set
. value .
append
. HTTP .
add
. () .
append .
unset
.
echo
. header .

```

```

    header .
add, unset . echo header . set,
. value , value
.

```

%t	epoch (1970 1 1) t=.
%D	...
%{FOOBAR}e	FOOBAR.

```

Header add, append, set
. env=... ( env=!...
) Header .
Header .

```



RequestHeader

:	HTTP
:	RequestHeader set append add unset <i>header</i> [<i>value</i>]
:	, , directory, .htaccess
Override :	FileInfo
:	Extension
:	mod_headers

HTTP , .

set

append

HTTP .

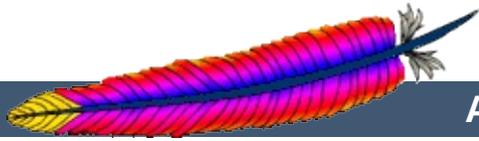
add

() .
append .

unset

append, set *value* . *value* .
unset *value* .

fixup **RequestHeader** .



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_imap

- ⋮ (imagemap)
- ⋮ Base
- ⋮ imap_module
- ⋮ mod_imap.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>
```

" .



```
#'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>
```



imap_base

```
base
ImapBase map|referer|URL
ImapBase http://servername/
, , directory, .htaccess
Override : Indexes
Base
mod_imap
```

```
ImapBase base .
base http://servername/.
```

- [UseCanonicalName](#)



imapDefault

```
:\n:\n      ImapDefault\n      error|nocontent|map|referer|URL\n:\n      ImapDefault nocontent\n:\n      , , directory, .htaccess\nOverride : Indexes\n:\n      Base\n:\n      mod_imap
```

ImapDefault default .
default . , default:
No Content nocontent. .



:	
:	ImapMenu none formatted semiformatted unformatte
:	, , directory, .htaccess
Override :	Indexes
:	Base
:	mod_imap

ImapMenu

none

ImapMenu none, default .

formatted

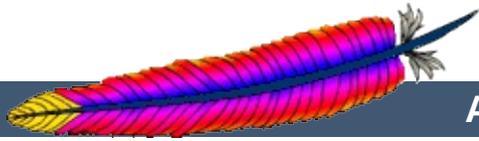
formatted .
 , .

semiformatted

semiformatted . HTML .
 , formatted .

unformatted

, .
 . , .
 .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_include

Description:	Server-parsed html documents (Server Side Includes)
Status:	Base
Module Identifier:	include_module
Source File:	mod_include.c
Compatibility:	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](#)

[International Customized Server Error Messages](#)

[Filters](#)

[SSI Tutorial](#)



Server Side Includes are implemented by the `INCLUDES` [filter](#). If documents containing server-side include directives are given the extension `.shtml`, the following directives will make Apache parse them and assign the resulting document the mime type of `text/html`:

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

The following directive must be given for the directories containing the `shtml` files (typically in a `<Directory>` section, but this directive is also valid in `.htaccess` files if `AllowOverride Options` is set):

```
Options +Includes
```

For backwards compatibility, the server-parsed [handler](#) also activates the `INCLUDES` filter. As well, Apache will activate the `INCLUDES` filter for any document with mime type `text/x-server-parsed-html` or `text/x-server-parsed-html3` (and the resulting output will have the mime type `text/html`).

For more information, see our [Tutorial on Server Side Includes](#).



Files processed for server-side includes no longer accept requests with PATH_INFO (trailing pathname information) by default. You can use the [AcceptPathInfo](#) directive to configure the server to accept requests with PATH_INFO.



The document is parsed as an HTML document, with special commands embedded as SGML comments. A command has the syntax:

```
<!--#element attribute=value attribute=value ... -->
```

The value will often be enclosed in double quotes, but single quotes (') and backticks (`) are also possible. Many commands only allow a single attribute-value pair. Note that the comment terminator (- ->) should be preceded by whitespace to ensure that it isn't considered part of an SSI token. Note that the leading <!--# is *one* token and may not contain any whitespaces.

The allowed elements are listed in the following table:

Element	Description
config	configure output formats
echo	print variables
exec	execute external programs
fsize	print size of a file
flastmod	print last modification time of a file
include	include a file
printenv	print all available variables
set	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:

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 be 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

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](#).

If you are matching positive (=), 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.
```

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:	String that ends an include element
Syntax:	SSIEndTag <i>tag</i>
Default:	SSIEndTag "-->"
Context:	server config, virtual host
Status:	Base
Module:	mod_include
Compatibility:	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](#)



Description:	Error message displayed when there is an SSI error
Syntax:	SSIErrorMsg <i>message</i>
Default:	SSIErrorMsg "[an error occurred while processing this directive]"
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Base
Module:	mod_include
Compatibility:	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 -->"
```



Description:	String that starts an include element
Syntax:	<code>SSIStartTag tag</code>
Default:	<code>SSIStartTag "<! - -#"</code>
Context:	server config, virtual host
Status:	Base
Module:	<code>mod_include</code>
Compatibility:	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](#)



Description:	Configures the format in which date strings are displayed
Syntax:	SSITimeFormat <i>formatstring</i>
Default:	SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Base
Module:	mod_include
Compatibility:	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".



Description:	String displayed when an unset variable is echoed
Syntax:	SSIUndefinedEcho <i>string</i>
Default:	SSIUndefinedEcho "(none)"
Context:	server config, virtual host
Status:	Base
Module:	mod_include
Compatibility:	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 -->"
```



Description:	Parse SSI directives in files with the execute bit set
Syntax:	XBitHack on off full
Default:	XBitHack off
Context:	server config, virtual host, directory, .htaccess
Override:	Options
Status:	Base
Module:	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

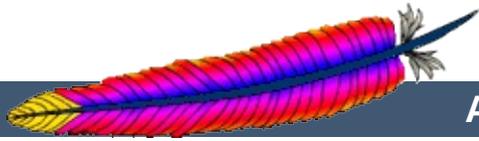
As for `on` but also test the group-execute bit. If it is set, then set the `Last-Modified` date of the returned file to be the last modified time of the file. If it is not set, then no last-modified date is sent. Setting this bit allows clients and proxies to cache the result of the request.

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).

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_info

```
┆  
┆  
┆ Extension  
┆ info_module  
┆ mod_info.c
```

mod_info httpd.conf .

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

<Location> <Limit> .

http://your.host.dom/server-info

```
,  
( User )  
mod_info, ( , .htaccess)  
.  
./,  
.
```

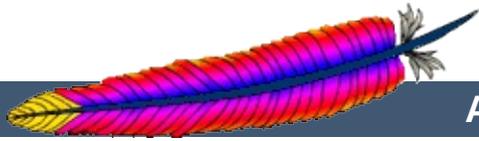


APACHE 2.0

```
server-info
AddModuleInfo module-name string
,
Extension
mod_info
1.3
```

module-name *string* HTML . ,

```
AddModuleInfo mod_auth.c 'See <a \
href="http://www.apache.org/docs/2.0/mod/mod_auth.html">\
http://www.apache.org/docs/2.0/mod/mod_auth.html</a>'
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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-isa .
ISAPI extension httpd.conf .
```

```
AddHandler isapi-isa .dll
```

```
. httpd.conf
.
```

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

```
ISAPI extension ISAPI extension
. , ISAPI .dll 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

:	ISAPI exntension	HSE_APPEND_LOG_PARAMETER
:	ISAPIAppendLogToErrors	on off
:	ISAPIAppendLogToErrors	off
:		, , directory, .htaccess
Override :	FileInfo	
:	Base	
:	mod_isapi	

ISAPI exntension HSE_APPEND_LOG_PARAMETER

.



ISAPIAppendLogToQuery

:	ISAPI exntension	HSE_APPEND_LOG_PARAMETER
:	ISAPIAppendLogToQuery	on off
:	ISAPIAppendLogToQuery	on
:		, , directory, .htaccess
Override :	FileInfo	
:	Base	
:	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 .



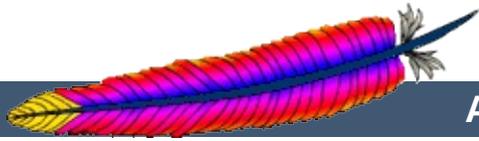
ISAPI extension (read ahead buffer)

```

: 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_ldap

Description:	LDAP connection pooling and result caching services for use by other LDAP modules
Status:	Experimental
Module Identifier:	ldap_module
Source File:	util_ldap.c
Compatibility:	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 support requires that [mod_ldap](#) be linked with one of the following LDAP SDKs: [OpenLDAP SDK](#) (both 1.x and 2.x), [Novell LDAP SDK](#) or the [iPlanet\(Netscape\) SDK](#).



Example Configuration

The following is an example configuration that uses `mod_ldap` to increase the performance of HTTP Basic authentication provided by `mod_auth_ldap`.

```
# Enable the LDAP connection pool and shared
# memory cache. Enable the LDAP cache status
# handler. Requires that mod_ldap and mod_auth_ldap
# be loaded. Change the "yourdomain.example.com" to
# match your domain.

LDAPSharedCacheSize 200000
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
    AuthLDAPEnabled on
    AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one
    AuthLDAPAuthoritative on
    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_auth_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 connections to an LDAP server is defined by the directives [LDAPTrustedCA](#) and [LDAPTrustedCAType](#). These directives specify the certificate file or database and the certificate type. Whenever the LDAP url includes *ldaps://*, [mod_ldap](#) will establish a secure connection to the LDAP server.

```
# Establish an SSL LDAP connection. Requires that
# mod_ldap and mod_auth_ldap be loaded. Change the
# "yourdomain.example.com" to match your domain.

LDAPTrustedCA /certs/certfile.der
LDAPTrustedCAType DER_FILE

<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPEnabled on
  AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
  AuthLDAPAuthoritative on
  Require valid-user
</Location>
```

If [mod_ldap](#) is linked against the Netscape/iPlanet LDAP SDK, it will not talk to any SSL server unless that server has a certificate signed by a known Certificate Authority. As part of the configuration [mod_ldap](#) needs to be told where it can find a database containing the known CAs. This database is in the same format as Netscape Communicator's `cert7.db` database. The easiest way to get this file is to start up a fresh copy of Netscape, and grab the resulting `$HOME/.netscape/cert7.db` file.



Description:	Maximum number of entries in the primary LDAP cache
Syntax:	LDAPCacheEntries <i>number</i>
Default:	LDAPCacheEntries 1024
Context:	server config
Status:	Experimental
Module:	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.



Description:	Time that cached items remain valid
Syntax:	LDAPCacheTTL <i>seconds</i>
Default:	LDAPCacheTTL 600
Context:	server config
Status:	Experimental
Module:	mod_ldap

Specifies the time (in seconds) that an item in the search/bind cache remains valid. The default is 600 seconds (10 minutes).



Description:	Specifies the socket connection timeout in seconds
Syntax:	LDAPConnectionTimeout <i>seconds</i>
Context:	server config
Status:	Experimental
Module:	mod_ldap

Specifies the timeout value (in seconds) in which the module will attempt to connect to the LDAP server. If a connection is not successful with the timeout period, either an error will be returned or the module will attempt to connect to a secondary LDAP server if one is specified. The default is 10 seconds.



Description: Number of entries used to cache LDAP compare operations

Syntax: LDAPOpCacheEntries *number*

Default: LDAPOpCacheEntries 1024

Context: server config

Status: Experimental

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.



Description:	Time that entries in the operation cache remain valid
Syntax:	LDAPOpCacheTTL <i>seconds</i>
Default:	LDAPOpCacheTTL 600
Context:	server config
Status:	Experimental
Module:	mod_ldap

Specifies the time (in seconds) that entries in the operation cache remain valid. The default is 600 seconds.



Description:	Sets the shared memory cache file
Syntax:	LDAPSharedCacheFile <i>directory-path/filename</i>
Context:	server config
Status:	Experimental
Module:	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.



Description:	Size in bytes of the shared-memory cache
Syntax:	LDAPSharedCacheSize <i>bytes</i>
Default:	LDAPSharedCacheSize 102400
Context:	server config
Status:	Experimental
Module:	mod_ldap

Specifies the number of bytes to allocate for the shared memory cache. The default is 100kb. If set to 0, shared memory caching will not be used.



Description:	Sets the file containing the trusted Certificate Authority certificate or database
Syntax:	LDAPTrustedCA <i>directory-path/filename</i>
Context:	server config
Status:	Experimental
Module:	mod_ldap

It specifies the directory path and file name of the trusted CA [mod_ldap](#) should use when establishing an SSL connection to an LDAP server. If using the Netscape/iPlanet Directory SDK, the file name should be `cert7.db`.



Description:	Specifies the type of the Certificate Authority file
Syntax:	LDAPTrustedCAType <i>type</i>
Context:	server config
Status:	Experimental
Module:	mod_ldap

The following types are supported:

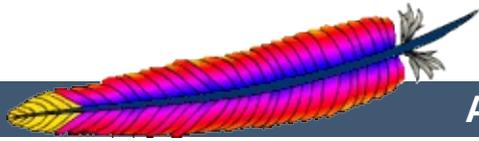
DER_FILE - file in binary DER format

BASE64_FILE - file in Base64 format

CERT7_DB_PATH - Netscape certificate database file ")

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_log_config

-
-
- Base
- log_config_module
- mod_log_config.c

• TransferLog , LogFormat ,
CustomLog . TransferLog CustomL



LogFormat CustomLog

C "\n" "\t"

" %" .

%%	(2.0.44)
%. . . a	IP-
%. . . A	() IP-
%. . . B	HTTP .
%. . . b	HTTP . CLF 0 ' -' .
%. . . {Foobar}C	<i>Foobar</i> .
%. . . D	().
%. . . {FOOBAR}e	<i>FOOBAR</i>
%. . . f	
%. . . h	
%. . . H	
%. . . {Foobar}i	<i>Foobar</i> : .
%. . . l	(identd) . IdentityCheck
%. . . m	
%. . . {Foobar}n	<i>Foobar (note)</i> .
%. . . {Foobar}o	<i>Foobar</i> : .

%...p	
%...P	ID.
%... {format}P	ID ID. format .(2.0.46)
%...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 . mo

```
"..." ( , "%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

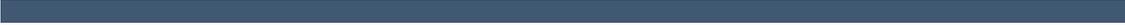
```
"%{Referer}i -> %U"
```

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
```

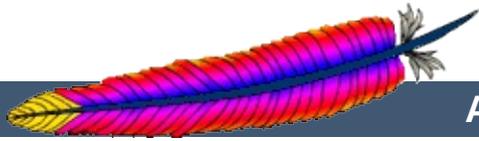


TransferLog

```
:\n:\nTransferLog file|pipe\n:\n,\n:\nBase\n:\nmod_log_config
```

```
.\n\nCustomLog\n\n(\n\n)\n\nLogFormat\n\nCommon\n\nLog 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_log_forensic`

Description:	Forensic Logging of the requests made to the server
Status:	Extension
Module Identifier:	<code>log_forensic_module</code>
Source File:	<code>mod_log_forensic.c</code>
Compatibility:	Available in version 2.0.50 and later

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.

This module was backported from version 2.1 which uses a more powerful APR version in order to generate the forensic IDs. If you want to run `mod_log_forensic` in version 2.0, you need to include `mod_unique_id` as well.

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.



Security Considerations

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.



ForensicLog Directive

Description:	Sets filename of the forensic log
Syntax:	ForensicLog <i>filename pipe</i>
Context:	server config, virtual host
Status:	Extension
Module:	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** takes the unique ID from **mod_unique_id**, so you need to load this module as well. (This requirement will not be necessary in version 2.1 and later, because of a more powerful APR version.) The ID token is attached to the request under the name `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

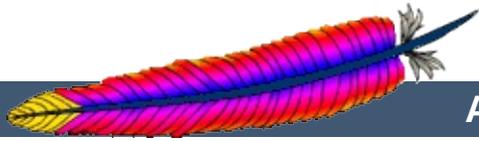
less privileged user.

Note

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

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_logio

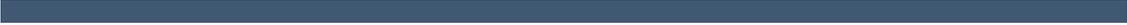
- ⋮
- ⋮ Extension
- ⋮ logio_module
- ⋮ mod_logio.c

SSL/TLS, SSL/TLS

mod_log_config.

mod_log_config

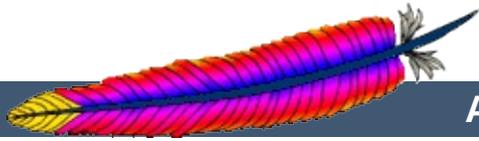




. " %"
:

Table with 2 columns and 2 rows containing alphanumeric strings and numbers.

:
:
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-agent}i\" %I %0"



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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



MCACHE_MAXOBJECTSIZE

```
MCACHE_MAXOBJECTSIZE  
: ( )  
: 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 .

```

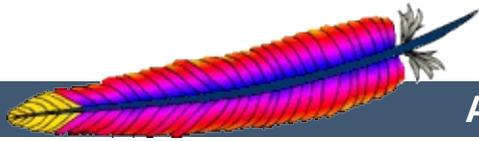
MCacheSize 700000

```

```

MCacheSize MCacheMaxObjectSize .

```



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_mime

Description:	Associates the requested filename's extensions with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding)
Status:	Base
Module Identifier:	mime_module
Source File:	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 which 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 by the module associated with the handler. For example, if the `.imap` extension is mapped to the handler `imap-file` (from `mod_imap`) 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_imap` imagemap file.



Content-Encoding

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.



Description:	Maps the given filename extensions to the specified content charset
Syntax:	<code>AddCharset <i>charset</i> <i>extension</i> [<i>extension</i>] ...</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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.

See also

- [mod_negotiation](#)
- [AddDefaultCharset](#)



Description:	Maps the given filename extensions to the specified encoding type
Syntax:	AddEncoding <i>MIME-enc extension</i> [<i>extension</i>] ...
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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.



Description:	Maps the filename extensions to the specified handler
Syntax:	<code>AddHandler <i>handler-name</i> <i>extension</i> [<i>extension</i>] ...</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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.

See also

- [SetHandler](#)



Description:	Maps filename extensions to the filters that will process client requests
Syntax:	<code>AddInputFilter <i>filter</i> [<i>;filter...</i>] <i>extension</i> [<i>extension</i>] ...</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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. Both the filter and *extension* arguments are case-insensitive, and the extension may be specified with or without a leading dot.

See also

- [RemoveInputFilter](#)
- [SetInputFilter](#)



Description:	Maps the given filename extension to the specified content language
Syntax:	AddLanguage <i>MIME-lang extension</i> [<i>extension</i>] ...
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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.

See also

- [mod_negotiation](#)



Description:	Maps filename extensions to the filters that will process responses from the server
Syntax:	<code>AddOutputFilter <i>filter</i> [;<i>filter</i>...] <i>extension</i> [<i>extension</i>] ...</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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. Both the *filter* and *extension* arguments are case-insensitive, and the extension may be specified with or without a leading dot.

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.

See also

- `DefaultType`
- `ForceType`



DefaultLanguage Directive

Description:	Sets all files in the given scope to the specified language
Syntax:	DefaultLanguage <i>MIME-lang</i>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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](#)



Description:	Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename
Syntax:	<code>ModMimeUsePathInfo On Off</code>
Default:	<code>ModMimeUsePathInfo Off</code>
Context:	directory
Status:	Base
Module:	<code>mod_mime</code>
Compatibility:	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](#)



Description:	The types of files that will be included when searching for a matching file with MultiViews
Syntax:	MultiViewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters]
Default:	MultiViewsMatch NegotiatedOnly
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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
```

See also

- [Options](#)
- [mod_negotiation](#)



Description:	Removes any character set associations for a set of file extensions
Syntax:	<code>RemoveCharset <i>extension</i> [<i>extension</i>] ...</code>
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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
```



Description:	Removes any content encoding associations for a set of file extensions
Syntax:	RemoveEncoding <i>extension</i> [<i>extension</i>] ...
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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.



Description:	Removes any handler associations for a set of file extensions
Syntax:	RemoveHandler <i>extension</i> [<i>extension</i>] ...
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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.



Description:	Removes any input filter associations for a set of file extensions
Syntax:	<code>RemoveInputFilter <i>extension</i> [<i>extension</i>] ...</code>
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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

Description:	Removes any language associations for a set of file extensions
Syntax:	RemoveLanguage <i>extension</i> [<i>extension</i>] ...
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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.



Description:	Removes any output filter associations for a set of file extensions
Syntax:	<code>RemoveOutputFilter <i>extension</i> [<i>extension</i>] ...</code>
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_mime
Compatibility:	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

Description:	Removes any content type associations for a set of file extensions
Syntax:	<code>RemoveType <i>extension</i> [<i>extension</i>] ...</code>
Context:	virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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

Description:	The location of the <code>mime.types</code> file
Syntax:	<code>TypesConfig file-path</code>
Default:	<code>TypesConfig conf/mime.types</code>
Context:	server config
Status:	Base
Module:	<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

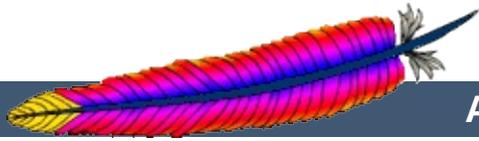
automatically rejected, as will any new two-letter extensions as they will likely conflict later with the already crowded language and character set namespace.

See also

- [mod_mime_magic](#)

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_mime_magic`

Description:	Determines the MIME type of a file by looking at a few bytes of its contents
Status:	Extension
Module Identifier:	<code>mime_magic_module</code>
Source File:	<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.

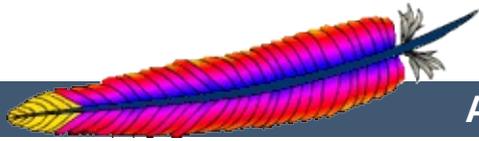


Description:	Enable MIME-type determination based on file contents using the specified magic file
Syntax:	MimeMagicFile <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	mod_mime_magic

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_negotiation

Description:	Provides for content negotiation
Status:	Base
Module Identifier:	negotiation_module
Source File:	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 0.0 to 1.0, 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

Description:	Allows content-negotiated documents to be cached by proxy servers
Syntax:	CacheNegotiatedDocs On Off
Default:	CacheNegotiatedDocs Off
Context:	server config, virtual host
Status:	Base
Module:	mod_negotiation
Compatibility:	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

Description:	Action to take if a single acceptable document is not found
Syntax:	ForceLanguagePriority None Prefer Fallback [Prefer Fallback]
Default:	ForceLanguagePriority Prefer
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_negotiation
Compatibility:	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](#)



Description:	The precedence of language variants for cases where the client does not express a preference
Syntax:	LanguagePriority <i>MIME-lang</i> [<i>MIME-lang</i>] ...
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	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
```

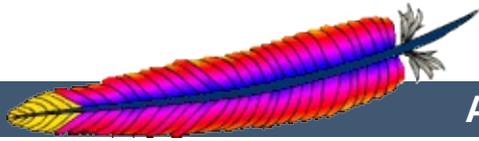
For a request for `foo.html`, where `foo.html.fr` and `foo.html.de` both existed, but the browser did not express a language preference, then `foo.html.fr` would be returned.

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_nw_ssl

Description:	Enable SSL encryption for NetWare
Status:	Base
Module Identifier:	nwssl_module
Source File:	mod_nw_ssl.c
Compatibility:	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.



Description:	List of additional client certificates
Syntax:	NWSSLTrustedCerts <i>filename</i> [<i>filename</i>] ...
Context:	server config
Status:	Base
Module:	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.



Description:	Allows a connection to be upgraded to an SSL connection upon request
Syntax:	NWSSLUpgradeable [<i>IP-address:</i>]portnumber
Context:	server config
Status:	Base
Module:	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.

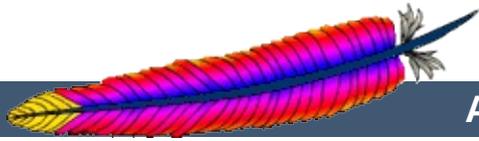


Description:	Enables SSL encryption for the specified port
Syntax:	<code>SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]</code>
Context:	server config
Status:	Base
Module:	mod_nw_ssl

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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_proxy`

Description:	HTTP/1.1 proxy/gateway server
Status:	Extension
Module Identifier:	<code>proxy_module</code>
Source File:	<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 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` 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_ssl](#)



Apache can be configured in both a *forward* and *reverse* proxy 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 and 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 proxys 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*, 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 name-space of the reverse proxy. The reverse proxy then decides where to send those requests, and returns the content as if it was 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](#).

Forward Proxy

```
ProxyRequests On
ProxyVia On
```

```
<Proxy *>
  Order deny,allow
  Deny from all
  Allow from internal.example.com
</Proxy>
```

Reverse Proxy

```
ProxyRequests Off
```

```
<Proxy *>
  Order deny,allow
  Allow from all
</Proxy>
```

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



Limiting access to your proxy

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_access](#).

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.



Why doesn't file type xxx download via FTP?

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
```

How can I force an FTP ASCII download of File xxx?

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 I access FTP files outside of my home 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 unsecure channels, an eavesdropper might intercept your password on its way.



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.



Proxy Requirements

For circumstances where you have a application server which doesn't implement keepalives or HTTP/1.1 properly, there are 2 environment variables which when set send a 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>
```



Description:	Ports that are allowed to CONNECT through the proxy
Syntax:	AllowCONNECT <i>port</i> [<i>port</i>] ...
Default:	AllowCONNECT 443 563
Context:	server config, virtual host
Status:	Extension
Module:	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.



Description:	Hosts, domains, or networks that will be connected to directly
Syntax:	NoProxy <i>host</i> [<i>host</i>] ...
Context:	server config, virtual host
Status:	Extension
Module:	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 two domains `.MyDomain.com` and `.mydomain.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.mit.edu
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 two hosts `WWW.MyDomain.com` and `www.mydomain.com.` (note the trailing period) are considered equal.

See also

- [DNS Issues](#)



Description:	Container for directives applied to proxied resources
Syntax:	<code><Proxy <i>wildcard-url</i>> ...</Proxy></code>
Context:	server config, virtual host
Status:	Extension
Module:	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>
```



ProxyBadHeader Directive

Description:	Determines how to handle bad header lines in a response
Syntax:	ProxyBadHeader IsError Ignore StartBody
Default:	ProxyBadHeader IsError
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in Apache 2.0.44 and later

The **ProxyBadHeader** directive determines the behaviour of **mod_proxy** if it receives syntactically invalid header lines (*i.e.* containing no colon). The following arguments are possible:

IsError

Abort the request and end up with a 502 (Bad Gateway) response. This is the default behaviour.

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

Description:	Default domain name for proxied requests
Syntax:	ProxyDomain <i>Domain</i>
Context:	server config, virtual host
Status:	Extension
Module:	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
```



Description:	Override error pages for proxied content
Syntax:	ProxyErrorOverride On Off
Default:	ProxyErrorOverride Off
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	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).



ProxyFtpDirCharset Directive

Description:	Define the character set for proxied FTP listings
Syntax:	ProxyFtpDirCharset <i>character set</i>
Default:	ProxyFtpDirCharset ISO-8859-1
Context:	server config, virtual host, directory
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in Apache 2.0.62 and later

The `ProxyFtpDirCharset` directive defines the character set to be set for FTP directory listings in HTML generated by [mod_proxy_ftp](#).



Description:	Determine size of internal data throughput buffer
Syntax:	ProxyIOBufferSize <i>bytes</i>
Default:	ProxyIOBufferSize 8192
Context:	server config, virtual host
Status:	Extension
Module:	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 less or equal 8192.

In almost every case there's no reason to change that value.



ProxyMatch Directive

Description:	Container for directives applied to regular-expression-matched proxied resources
Syntax:	<code><ProxyMatch regex> ...</ProxyMatch></code>
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `<ProxyMatch>` directive is identical to the `<Proxy>` directive, except it matches URLs using regular expressions.



ProxyMaxForwards Directive

Description:	Maximum number of proxies that a request can be forwarded through
Syntax:	ProxyMaxForwards <i>number</i>
Default:	ProxyMaxForwards 10
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	Available in Apache 2.0 and later

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 is set to prevent infinite proxy loops, or a DoS attack.

Example

```
ProxyMaxForwards 15
```



ProxyPass Directive

Description:	Maps remote servers into the local server URL-space
Syntax:	ProxyPass [<i>path</i>] ! <i>url</i>
Context:	server config, virtual host, directory
Status:	Extension
Module:	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. *path* is the name of a local virtual path; *url* is a partial URL for the remote server and cannot include a query string.

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`.

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`.

Note

Order is important. you need to put the exclusions *before* the general proxypass directive.

When used inside a [<Location>](#) section, the first argument is omitted and the local directory is obtained from the [<Location>](#).

The [ProxyRequests](#) directive should usually be set **off** when using [ProxyPass](#).

If you require a more flexible reverse-proxy configuration, see the [RewriteRule](#) directive with the [P] flag.



Description:	Adjusts the URL in HTTP response headers sent from a reverse proxied server
Syntax:	ProxyPassReverse [<i>path</i>] <i>url</i>
Context:	server config, virtual host, directory
Status:	Extension
Module:	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 to avoid by-passing 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 by-pass 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 - the same way they are used 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/
```

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 ProxyPass provides here). It also takes care of redirects the server `backend.example.com` sends: when `http://backend.example.com/bar` is redirected by him 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 pass-through feature (`RewriteRule . . . [P]`) from [mod_rewrite](#) because its doesn't depend on a corresponding [ProxyPass](#) directive.

When used inside a [<Location>](#) section, the first argument is omitted and the local directory is obtained from the [<Location>](#).



ProxyPreserveHost Directive

Description:	Use incoming Host HTTP request header for proxy request
Syntax:	ProxyPreserveHost On Off
Default:	ProxyPreserveHost Off
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	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 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
```



ProxyRemote Directive

Description:	Remote proxy used to handle certain requests
Syntax:	ProxyRemote <i>match remote-server</i>
Context:	server config, virtual host
Status:	Extension
Module:	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 is supported by this module.

Example

```
ProxyRemote http://goodguys.com/ http://mirrorguys.com:8000  
ProxyRemote * http://cleversite.com  
ProxyRemote ftp http://ftpproxy.mydomain.com: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

Description:	Remote proxy used to handle requests matched by regular expressions
Syntax:	<code>ProxyRemoteMatch <i>regex remote-server</i></code>
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy

The `ProxyRemoteMatch` is identical to the `ProxyRemote` directive, except the first argument is a regular expression match against the requested URL.



ProxyRequests Directive

Description:	Enables forward (standard) proxy requests
Syntax:	ProxyRequests On Off
Default:	ProxyRequests Off
Context:	server config, virtual host
Status:	Extension
Module:	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 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.

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.



Description:	Network timeout for proxied requests
Syntax:	ProxyTimeout <i>seconds</i>
Default:	ProxyTimeout 300
Context:	server config, virtual host
Status:	Extension
Module:	mod_proxy
Compatibility:	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.



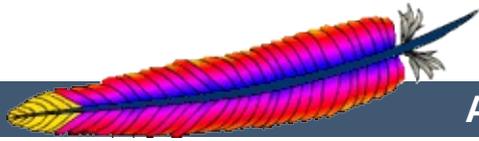
Description:	Information provided in the <code>Via</code> HTTP response header for proxied requests
Syntax:	<code>ProxyVia On Off Full Block</code>
Default:	<code>ProxyVia Off</code>
Context:	server config, virtual host
Status:	Extension
Module:	<code>mod_proxy</code>

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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_proxy_connect`

Description:	<code>mod_proxy</code> extension for CONNECT request handling
Status:	Extension
Module Identifier:	<code>proxy_connect_module</code>
Source File:	<code>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.

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.

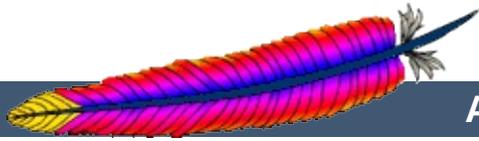
See also

[AllowCONNECT](#)

[mod_proxy](#)

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_proxy_ftp`

Description:	FTP support module for mod_proxy
Status:	Extension
Module Identifier:	<code>proxy_ftp_module</code>
Source File:	<code>proxy_ftp.c</code>

Summary

This module *requires* the service of [mod_proxy](#). It provides support for the proxying FTP sites.

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

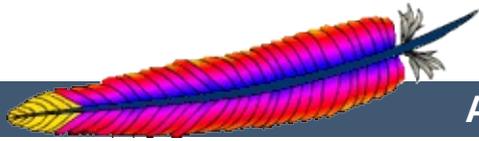
Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

See also

[mod_proxy](#)

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_proxy_http`

Description:	HTTP support module for mod_proxy
Status:	Extension
Module Identifier:	<code>proxy_http_module</code>
Source File:	<code>proxy_http.c</code>

Summary

This module *requires* the service of [mod_proxy](#). It provides the features used for proxying HTTP 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.

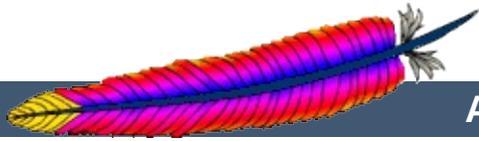
See also

[mod_proxy](#)

[mod_proxy_connect](#)

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_rewrite`

Description:	Provides a rule-based rewriting engine to rewrite requested URLs on the fly
Status:	Extension
Module Identifier:	<code>rewrite_module</code>
Source File:	<code>mod_rewrite.c</code>
Compatibility:	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](#)



Apache processes a HTTP request in several 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 (used after the HTTP request has been read, but before any authorization starts) and the Fixup hook (triggered after the authorization phases, and after the per-directory config files (`.htaccess`) have been read, but before the content handler is activated).

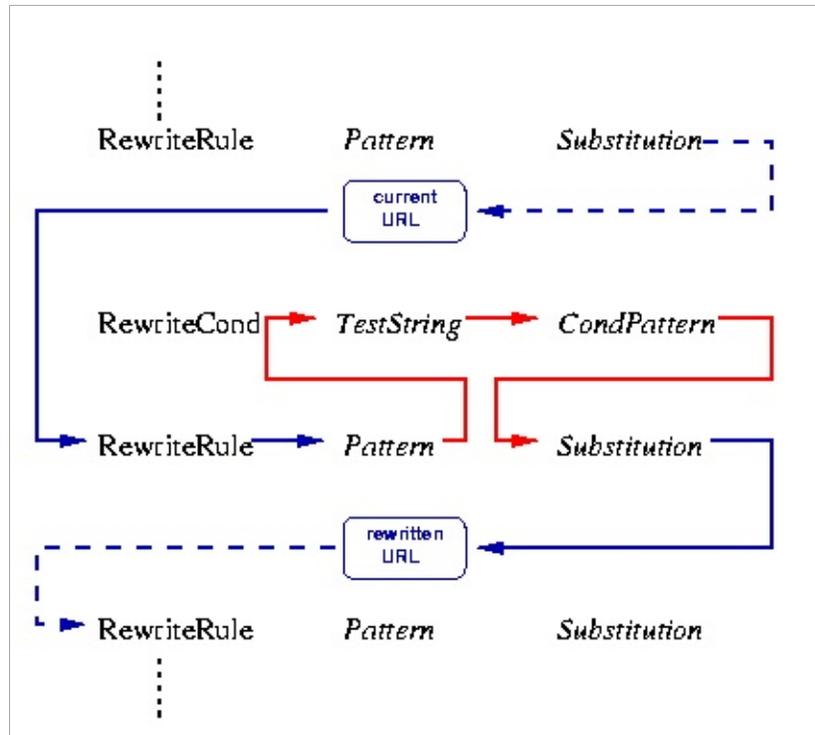
Once a request comes in, and Apache has determined the appropriate server (or virtual server), the rewrite engine starts the URL-to-filename translation, processing the `mod_rewrite` directives from the per-server configuration. A few steps later, when the final data directories are found, the per-directory configuration directives of `mod_rewrite` are triggered in the Fixup phase.



URL Rewriting

When `mod_rewrite` is triggered during these two API phases, it reads the relevant rulesets from its configuration structure (which was either created on startup, for per-server context, or during the directory traversal for per-directory context). The URL rewriting engine is started with the appropriate ruleset (one or more rules together with their conditions), and its operation 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 rewrite engine processes them in a particular (not always obvious) order, as follows: The rewrite engine loops through the rulesets (each ruleset being made up of `RewriteRule` directives, with or without `RewriteConds`), rule by rule. When a particular rule is matched, `mod_rewrite` also checks the corresponding conditions (`RewriteCond` directives). For historical reasons the conditions are given first, making the control flow a little bit long-winded. See Figure 1 for more details.



Figure

1: The control flow of the rewrite engine through a rewrite ruleset

As above, first the URL is matched against the *Pattern* of a rule. If it does not match, [mod_rewrite](#) immediately stops processing that rule, and goes on to the next rule. If the *Pattern* matches, [mod_rewrite](#) checks for rule conditions. If none are present, the URL will be replaced with a new string, constructed from the *Substitution* string, and [mod_rewrite](#) goes on to the next rule.

If **RewriteConds** exist, an inner loop is started, processing them in the order that they are listed. Conditions are not matched against the current URL directly. A *TestString* is constructed by expanding variables, back-references, map lookups, etc., against which the *CondPattern* is matched. If the pattern fails to match one of the conditions, the complete set of rule and associated conditions fails. If the pattern matches a given condition, then matching continues to the next condition, until no more conditions are available. If all conditions match, processing is continued with

the substitution of the *Substitution* string for the URL.



Regular Back-Reference Availability

Using parentheses in *Pattern* or in one of the *CondPatterns* causes back-references to be internally created. These can later be referenced using the strings $\$N$ and $\%N$ (see below), for creating the *Substitution* and *TestString* strings. Figure 2 attempts to show how the back-references are transferred through the process for later expansion.

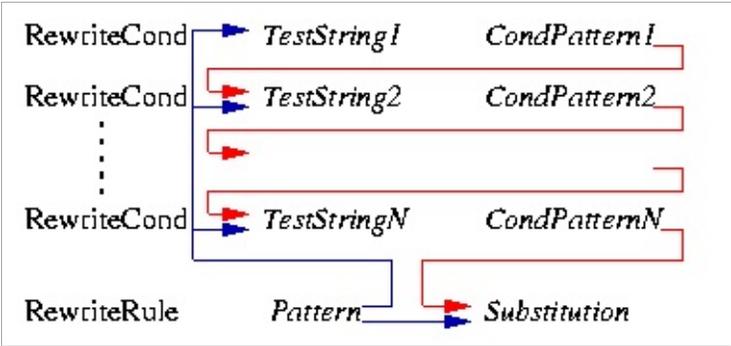


Figure 2: The back-reference flow through a rule.



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/
```



For numerous examples of common, and not-so-common, uses for `mod_rewrite`, see the [Rewrite Guide](#), and the [Advanced Rewrite Guide](#) documents.



Description:	Sets the base URL for per-directory rewrites
Syntax:	RewriteBase <i>URL-path</i>
Default:	See usage for information.
Context:	directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	mod_rewrite

The `RewriteBase` directive explicitly sets the base URL for per-directory rewrites. As you will see below, `RewriteRule` can be used in per-directory config files (`.htaccess`). In such a case, it will act locally, stripping the local directory prefix before processing, and applying rewrite rules only to the remainder. When processing is complete, the prefix is automatically added back to the path. The default setting is; `RewriteBase physical-directory-path`

When a substitution occurs for a new URL, this module has to re-inject the URL into the server processing. To be able to do this it needs to know what the corresponding URL-prefix or URL-base is. By default this prefix is the corresponding filepath itself. **However, for most websites, URLs are NOT directly related to physical filename paths, so this assumption will often be wrong!** Therefore, you can use the `RewriteBase` directive to specify the correct URL-prefix.

If your webserver's URLs are **not** directly related to physical file paths, you will need to use `RewriteBase` in every `.htaccess` file where you want to use `RewriteRule` directives.

For example, assume the following per-directory config file:

```

#
# /abc/def/.htaccess -- per-dir config file for directory /abc/c
# Remember: /abc/def is the physical path of /xyz, i.e., the ser
#         has a 'Alias /xyz /abc/def' directive e.g.
#
RewriteEngine On

# let the server know that we were reached via /xyz and not
# via the physical path prefix /abc/def
RewriteBase /xyz

# now the rewriting rules
RewriteRule ^oldstuff\.html$ newstuff.html

```

In the above example, a request to `/xyz/oldstuff.html` gets correctly rewritten to the physical file `/abc/def/newstuff.html`.

For Apache Hackers

The following list gives detailed information about the internal processing steps:

Request:

```
/xyz/oldstuff.html
```

Internal Processing:

```

/xyz/oldstuff.html -> /abc/def/oldstuff.html (per-server /
/abc/def/oldstuff.html -> /abc/def/newstuff.html (per-dir F
/abc/def/newstuff.html -> /xyz/newstuff.html (per-dir F
/xyz/newstuff.html -> /abc/def/newstuff.html (per-server /

```

Result:

```
/abc/def/newstuff.html
```

This seems very complicated, but is in fact correct Apache internal processing. Because the per-directory rewriting comes late in the process, the rewritten request has to be re-injected into the Apache kernel, as if it were a new request. (See [mod_rewrite technical details](#).) This is not the serious overhead it may seem to be - this re-injection is completely internal to the Apache server (and the same procedure is used by many other

operations within Apache).



Description:	Defines a condition under which rewriting will take place
Syntax:	<code>RewriteCond <i>TestString CondPattern</i></code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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 &

	request:	
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	
server internals:	system stuff:	special:
DOCUMENT_ROOT	TIME_YEAR	API_VERSION
SERVER_ADMIN	TIME_MON	THE_REQUEST
SERVER_NAME	TIME_DAY	REQUEST_URI
SERVER_ADDR	TIME_HOUR	REQUEST_METHOD
SERVER_PORT	TIME_MIN	IS_SUBREQUEST
SERVER_PROTOCOL	TIME_SEC	HTTP_PROTOCOL
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. 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.

REQUEST_URI

The resource requested in the HTTP request line. (In the example above, this would be "/index.html".)

REQUEST_FILENAME

The full local filesystem path to the file or script matching the request.

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).

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.
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`.
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 **directory**)
Treats the *TestString* as a pathname and tests whether or

not it exists, and is a directory.

- **'-f'** (is regular file)
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file.
- **'-s'** (is regular file, with size)
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file with size greater than zero.
- **'-l'** (is symbolic link)
Treats the *TestString* as a pathname and tests whether or not it exists, and is a symbolic link.
- **'-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!

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.

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.



Runtime Engine Directive

Description:	Enables or disables runtime rewriting engine
Syntax:	RewriteEngine on off
Default:	RewriteEngine off
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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, by default, rewrite configurations are not inherited. This means that you need to have a `RewriteEngine on` directive for each virtual host in which you wish to use it.

`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`



Description:	Sets the name of the lock file used for RewriteMap synchronization
Syntax:	RewriteLock <i>file-path</i>
Context:	server config
Status:	Extension
Module:	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

Description:	Sets the name of the file used for logging rewrite engine processing
Syntax:	RewriteLog <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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!`

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

```
RewriteLog "/usr/local/var/apache/logs/rewrite.log"
```



Description:	Sets the verbosity of the log file used by the rewrite engine
Syntax:	<code>RewriteLogLevel <i>Level</i></code>
Default:	<code>RewriteLogLevel 0</code>
Context:	server config, virtual host
Status:	Extension
Module:	<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
```



Description:	Defines a mapping function for key-lookup
Syntax:	<code>RewriteMap MapName MapType:MapSource</code>
Context:	server config, virtual host
Status:	Extension
Module:	mod_rewrite
Compatibility:	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.

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 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.

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. You can create such a file with any DBM tool or with the following Perl script. Be sure to adjust it to create the appropriate type of DBM. The example creates an NDBM

file.

```
#!/path/to/bin/perl
##
##  txt2dbm -- convert txt map to dbm format
##

use NDBM_File;
use Fcntl;

($txtmap, $dbmmap) = @ARGV;

open(TXT, "<$txtmap") or die "Couldn't open $txtmap!\n";
tie (%DB, 'NDBM_File', $dbmmap, O_RDWR|O_TRUNC|O_CREAT, 0644)
    or die "Couldn't create $dbmmap!\n";

while (<TXT>) {
    next if (/^\s*#/ or /^\s*$/);
    $DB{$1} = $2 if (/^\s*(\S+)\s+(\S+)/);
}

untie %DB;
close(TXT);
```

```
$ txt2dbm map.txt map.db
```

- **Internal Function**

MapType: `int`, MapSource: Internal Apache function

Here, the source is an internal Apache function. Currently you cannot create your own, but the following functions already exist:

- **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!



Description:	Sets some special options for the rewrite engine
Syntax:	<code>RewriteOptions Options</code>
Default:	<code>RewriteOptions MaxRedirects=10</code>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	mod_rewrite
Compatibility:	MaxRedirects is available in Apache 2.0.45 and later

The `RewriteOptions` directive sets some special options for the current per-server or per-directory configuration. The *Option* strings can 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.

MaxRedirects=number

In order to prevent endless loops of internal redirects issued by per-directory `RewriteRules`, `mod_rewrite` aborts the request after reaching a maximum number of such redirects and responds with an 500 Internal Server Error. If you really need more internal redirects than 10 per request, you may increase the default to the desired value.

AllowAnyURI

When `RewriteRule` is used in `VirtualHost` or server context with version 2.0.65 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`. This flag is available for Apache HTTP Server 2.0.65 and later.



Description:	Defines rules for the rewriting engine
Syntax:	RewriteRule <i>Pattern Substitution</i>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	mod_rewrite
Compatibility:	The cookie-flag is available in Apache 2.0.40 and later.

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, which is applied to the current URL. ``Current" means the value of the URL when this rule is applied. This may not be the originally requested URL, which may already have matched a previous rule, and have been altered.

Some hints on the syntax of regular expressions:

Text:	
.	Any single character
[chars]	Character class: Any character of the class ``chars
[^chars]	Character class: Not a character of the class ``cha
text1 text2	Alternative: text1 or text2
Quantifiers:	
?	0 or 1 occurrences of the preceding text
*	0 or N occurrences of the preceding text (N > 0)
+	1 or N occurrences of the preceding text (N > 1)
Grouping:	
(text)	Grouping of text

(used either to set the borders of an alternative or to make backreferences, where the **N**th group can be referred to on the RHS of a RewriteRule as **\$N**)

Anchors:

^ Start-of-line anchor
\$ End-of-line anchor

Escaping:

\char escape the given char
(for instance, to specify the chars ".[]()" etc.)

For more information about regular expressions, have a look at the perl regular expression manpage ("[perldoc perlre](#)"). If you are interested in more detailed information about regular expressions and their variants (POSIX regex etc.) the following book is dedicated to this topic:

Mastering Regular Expressions, 2nd Edition

Jeffrey E.F. Friedl

O'Reilly & Associates, Inc. 2002

ISBN 0-596-00289-0

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 which is substituted

for (or replaces) the original URL which *Pattern* matched. In addition to plain text, it can include

1. back-references ($\$N$) to the RewriteRule pattern
2. back-references ($\%N$) to the last matched RewriteCond pattern
3. server-variables as in rule condition test-strings ($\% \{VARNAME\}$)
4. [mapping-function](#) calls ($\$\{mapname : key | default\}$)

Back-references are identifiers of the form $\$N$ ($N=0..9$), which will be replaced by the contents of the N th group of the matched *Pattern*. The server-variables are the same as for the *TestString* of a RewriteCond directive. The mapping-functions come from the RewriteMap directive and are explained there. These three types of variables are expanded in the order above.

As already mentioned, all rewrite rules are applied to the *Substitution* (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 - see below.

There is a special substitution string named '-' which means: **NO substitution!** This is useful in providing rewriting rules which **only** match URLs but do not substitute anything for them. It is commonly used in conjunction with the **C** (chain) flag, in order to apply more than one pattern before substitution occurs.

Additionally you can set special flags for *Substitution* by appending **[flags]** as the third argument to the RewriteRule directive. *Flags* is a comma-separated list of any of the following flags:

- **'chain|C'** (chained with next rule)

This flag chains the current rule with the next rule (which itself can be chained with the following rule, and so on). This has the following effect: if a rule matches, then processing continues as usual - the flag has no effect. If the rule does **not** match, then all following chained rules are skipped. For instance, it can be used to remove the ``.www" part, inside a per-directory rule set, when you let an external redirect happen (where the ``.www" part should not occur!).

- '**cookie** | **CO**=*NAME*:*VAL*:*domain*[:*lifetime*[:*path*]]' (set **cookie**)
This sets a cookie in the client's browser. The cookie's name is specified by *NAME* and the value is *VAL*. The *domain* field is the domain of the cookie, such as '.apache.org', the optional *lifetime* is the lifetime of the cookie in minutes, and the optional *path* is the path of the cookie
- '**env** | **E**=*VAR*:*VAL*' (set **environment variable**)
This forces an environment variable named *VAR* to be set to the value *VAL*, where *VAL* can contain regexp backreferences (*\$N* and *%N*) which will be expanded. You can use this flag more than once, to set more than one variable. The variables can later be dereferenced in many situations, most commonly from within XSSI (via `<! - -#echo var="VAR" - ->`) or CGI (`$ENV{ 'VAR' }`). You can also dereference the variable in a later RewriteCond pattern, using `%{ENV:VAR}`. Use this to strip information from URLs, while maintaining a record of that information.
- '**forbidden** | **F**' (force URL to be **forbidden**)
This forces the current URL to be forbidden - it immediately sends back a HTTP response of 403 (FORBIDDEN). Use this flag in conjunction with appropriate RewriteConds to conditionally block some URLs.
- '**gone** | **G**' (force URL to be **gone**)
This forces the current URL to be gone - it immediately sends back a HTTP response of 410 (GONE). Use this flag to mark

pages which no longer exist as gone.

- **'last | L'** (last rule)

Stop the rewriting process here and don't apply any more rewrite rules. This corresponds to the Perl `last` command or the `break` command in C. Use this flag to prevent the currently rewritten URL from being rewritten further by following rules. For example, use it to rewrite the root-path URL `('/')` to a real one, e.g., `'/e/www/'`.

- **'next | N'** (next round)

Re-run the rewriting process (starting again with the first rewriting rule). This time, the URL to match is no longer the original URL, but rather the URL returned by the last rewriting rule. This corresponds to the Perl `next` command or the `continue` command in C. Use this flag to restart the rewriting process - to immediately go to the top of the loop.

Be careful not to create an infinite loop!

- **'nocase | NC'** (no case)

This makes the *Pattern* case-insensitive, ignoring difference between 'A-Z' and 'a-z' when *Pattern* is matched against the current URL.

- **'noescape | NE'** (no URI escaping of output)

This flag prevents `mod_rewrite` from applying the usual URI escaping rules to the result of a rewrite. Ordinarily, special characters (such as '%', '\$', ';', and so on) will be escaped into their hexcode equivalents ('%25', '%24', and '%3B', respectively); this flag prevents this from happening. This allows percent symbols to appear in the output, as in

```
RewriteRule /foo/(.*) /bar?arg=P1\%3d$1 [R,NE]
```

which would turn `'/foo/zed'` into a safe request for `'/bar?arg=P1=zed'`.

- **'nosubreq | NS'** (not for internal sub-requests)

This flag forces the rewrite engine to skip a rewrite rule if the current request is an internal sub-request. For instance, sub-requests occur internally in Apache when [mod_include](#) tries to find out information about possible directory default files (`index.xxx`). On sub-requests it is not always useful, and can even cause errors, if the complete set of rules are applied. Use this flag to exclude some 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.

- **'proxy|P'** (force proxy)

This flag forces the substitution part to be internally sent as a proxy request and immediately (rewrite processing stops here) put through the [proxy module](#). You must make sure that the substitution string is a valid URI (typically starting with `http://hostname`) which can be handled by the Apache proxy module. 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.

Note: [mod_proxy](#) must be enabled in order to use this flag.

- **'passthrough|PT'** (pass through to next handler)

This flag forces the rewrite engine to set the `uri` field of the internal `request_rec` structure to the value of the `filename` field. This flag is just a hack to enable post-processing of the output of `RewriteRule` directives, using `Alias`, `ScriptAlias`, `Redirect`, and other directives from various URI-to-filename translators. For example, to rewrite `/abc` to `/def` using [mod_rewrite](#), and then `/def` to `/ghi` using [mod_alias](#):

```
RewriteRule ^/abc(.*) /def$1 [PT]
```

```
Alias /def /ghi
```

If you omit the PT flag, `mod_rewrite` will rewrite `uri=/abc/...` to `filename=/def/...` as a full API-compliant URI-to-filename translator should do. Then `mod_alias` will try to do a URI-to-filename transition, which will fail.

Note: You must use this flag if you want to mix directives from different modules which allow URL-to-filename translators. The typical example is the use of `mod_alias` and `mod_rewrite`.

- **'qsappend|QSA'** (query string append)
This flag forces the rewrite engine to append a query string part of the substitution string to the existing string, instead of replacing it. Use this when you want to add more data to the query string via a rewrite rule.
- **'redirect|R [=code]'** (force redirect)
Prefix *Substitution* with `http://thishost[:thisport]/` (which makes the new URL a URI) to force an external redirection. If no *code* is given, a HTTP response of 302 (MOVED TEMPORARILY) will be returned. If you want to use other response codes in the range 300-400, simply specify the appropriate number or use one of the following symbolic names: `temp` (default), `permanent`, `seeother`. Use this for rules to canonicalize the URL and return it to the client - to translate `~/~` into `~/u/`, or to always append a slash to `/u/user`, etc.
Note: When you use this flag, make sure that the substitution field is a valid URL! Otherwise, you will be redirecting to an invalid location. Remember that this flag on its own will only prepend `http://thishost[:thisport]/` to the URL, and rewriting will continue. Usually, you will want to stop rewriting at this point, and redirect immediately. To stop rewriting, you

should add the 'L' flag.

- **'skip|S=num'** (skip next rule(s))

This flag forces the rewriting engine to skip the next *num* rules in sequence, if the current rule matches. 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. (This is **not** the same as the 'chain|C' flag!)

- **'type|T=MIME-type'** (force MIME type)

Force the MIME-type of the target file to be *MIME-type*. This can be used to set up the content-type based on some conditions. For example, the following snippet allows .php files to be *displayed* by `mod_php` if they are called with the .phps extension:

```
RewriteRule ^(.+\.\php)s$ $1 [T=application/x-httpd-php-source]
```

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.

Note: Enabling rewrites in per-directory context

To enable the rewriting engine for per-directory configuration files, you need to set ```RewriteEngine On``` in these files **and** ```Options FollowSymLinks``` must be enabled. If your administrator has disabled override of `FollowSymLinks` for a user's directory, then you cannot use the rewriting engine. This restriction is needed for security reasons.

Note: Pattern matching in per-directory context

Never forget that *Pattern* is applied to a complete URL in per-server configuration files. **However, in per-directory configuration files, the per-directory prefix (which always is the same for a specific directory) is automatically *removed* for the pattern matching and automatically *added* after the substitution has been done.** This feature is essential for many sorts of rewriting - without this, you would always have to match the parent directory which is not always possible.

There is one exception: If a substitution string starts with ```http://```, then the directory prefix will **not** be added, and an external redirect or proxy throughput (if flag **P** is used) is forced!

Note: Substitution of Absolute URLs

When you prefix a substitution field with `http://thishost[:thisport]`, `mod_rewrite` will automatically strip that out. This auto-reduction on URLs with an implicit external redirect is most useful in combination with a mapping-function which generates the hostname part.

Remember: An unconditional external redirect to your own server will not work with the prefix `http://thishost` because of this feature. To achieve such a self-redirect, you have to use the **R**-flag.

Note: Query String

The *Pattern* will not be matched against the query string. Instead, you must use a `RewriteCond` with the `%{QUERY_STRING}` variable. 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 a new query string with an old one, use the [QSA] flag.

Here are all possible substitution combinations and their meanings:

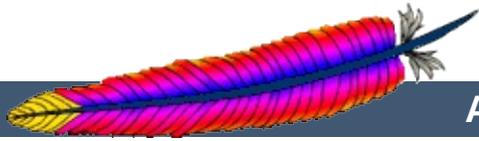
**Inside per-server configuration (httpd.conf)
for request `GET /somepath/pathinfo`:**

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 pro

**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 pro

Copyright 2013 The Apache Software Foundation.
 Licensed under the [Apache License, Version 2.0](#).



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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
:      1.2 ( 1.2                                mod_browse
      )

```

[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] . " ()

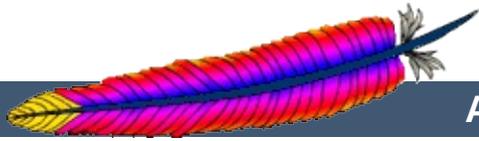
(regex) [Perl](#) . POSIX.2 egrep . regex


```
SetEnvIfNoCase attribute regex [!]env-  
variable[=value] [[!]env-  
variable[=value]] ...  
    , , directory, .htaccess  
Override : FileInfo  
    Base  
    mod_setenvif  
    1.3
```

SetEnvIfNoCase **SetEnvIf** , .
:

```
SetEnvIfNoCase Host Apache\.Org site=apache
```

HTTP Host: Apache.Org, apache.org
site " apache".



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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
```



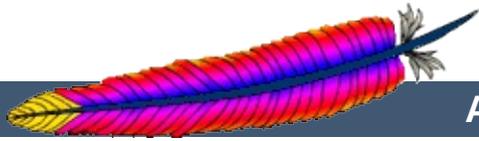
LoadModule

```
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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_speling

⋮

URL

⋮

Extension

⋮

speling_module

⋮

mod_speling.c

.)

,

• , "document not

• "" ,

• ,

found ()" .



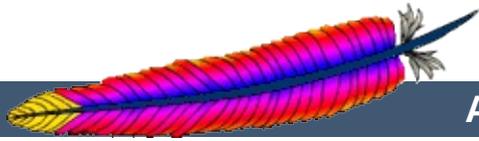
CheckSpelling

```

:
:      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" .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_ssl

Description:	Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols
Status:	Extension
Module Identifier:	ssl_module
Source File:	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 provides a lot of SSL information as additional environment variables to the SSI and CGI namespace. The generated variables are listed in the table below. For backward compatibility the information can be made available under different names, too. Look in the [Compatibility](#) chapter for details on the compatibility variables.

Variable Name:	Value Type:	Description:
HTTPS	flag	HTTPS is being used.
SSL_PROTOCOL	string	The SSL protocol version (SSLv2, SSLv3, TLSv1)
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_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_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

[where x509 is a component of a X.509 DN:
C, ST, L, O, OU, CN, T, I, G, S, D, UID, Email]



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"
```



Description:	File of concatenated PEM-encoded CA Certificates for Client Auth
Syntax:	<code>SSLCACertificateFile</code> <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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
```



Description:	Directory of PEM-encoded CA Certificates for Client Auth
Syntax:	SSLCACertificatePath <i>directory-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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/
```



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
```



Description:	Directory of PEM-encoded CA CRLs for Client Auth
Syntax:	SSLCARevocationPath <i>directory-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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/
```



Description:	File of PEM-encoded Server CA Certificates
Syntax:	<code>SSLCertificateChainFile</code> <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	<code>mod_ssl</code>

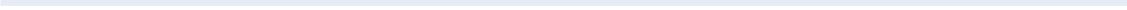
This directive sets the optional *all-in-one* file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate. Such 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. That's usually not one expect.

But be careful: Providing the certificate chain works only if you are using a *single* (either 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
```



Description:	Server PEM-encoded X.509 Certificate file
Syntax:	SSLCertificateFile <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl

This directive points to the PEM-encoded Certificate file for the server and optionally also to the corresponding RSA or DSA Private Key file for it (contained in the same file). If the contained Private Key is encrypted the Pass Phrase dialog is forced at startup time. This directive can be used up to two times (referencing different filenames) when both a RSA and a DSA based server certificate is used in parallel.

Example

```
SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
```



Description:	Server PEM-encoded Private Key file
Syntax:	SSLCertificateKeyFile <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl

This directive points to the PEM-encoded Private Key file for the server. If the Private Key is not combined with the Certificate in the `SSLCertificateFile`, use this additional directive to point to the file with the stand-alone Private Key. When `SSLCertificateFile` is used and the file contains both the Certificate and the Private Key this directive need not be used. But we strongly discourage this practice. Instead we recommend you to separate the Certificate and the Private Key. If the contained Private Key is encrypted, the Pass Phrase dialog is forced at startup time. This directive can be used up to two times (referencing different filenames) when both a RSA and a DSA based private key is used in parallel.

Example

```
SSLCertificateKeyFile  
/usr/local/apache2/conf/ssl.key/server.key
```



Description:	Cipher Suite available for negotiation in SSL handshake
Syntax:	SSLCipherSuite <i>cipher-spec</i>
Default:	SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
Context:	server config, virtual host, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	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](#)).

Tag	Description
<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)

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



Description:	SSL Engine Operation Switch
Syntax:	SSLEngine on off
Default:	SSLEngine off
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for a particular 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>
```



Description:	Option to prefer the server's cipher preference order
Syntax:	SSLHonorCipherOrder <i>flag</i>
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl
Compatibility:	Available in Apache 2.0.65 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
```



Description:	Option to enable support for insecure renegotiation
Syntax:	SSLInsecureRenegotiation <i>flag</i>
Default:	SSLInsecureRenegotiation off
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl
Compatibility:	Available in httpd 2.0.64 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.



Description:	Semaphore for internal mutual exclusion of operations
Syntax:	SSLMutex <i>type</i>
Default:	SSLMutex none
Context:	server config
Status:	Extension
Module:	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
```



Description:	Configure various SSL engine run-time options
Syntax:	SSLOptions [+ -] <i>option</i> ...
Context:	server config, virtual host, directory, .htaccess
Override:	Options
Status:	Extension
Module:	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.
- `CompatEnvVars`
When this option is enabled, additional CGI/SSI environment variables are created for backward compatibility to other Apache SSL solutions. Look in the [Compatibility](#) chapter for details on the particular variables generated.
- `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 $n = 0,1,2,\dots$). These contain the PEM-encoded X.509 Certificates of server and client for the current HTTPS connection and can be used by CGI scripts for deeper Certificate checking. Additionally all other certificates of the client certificate chain are provided, too. This bloats up the environment a little bit which is why you have to use this option to enable it on demand.

- **FakeBasicAuth**

When this option is enabled, the Subject Distinguished Name (DN) of the Client X509 Certificate is translated into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is just the Subject of the Client's X509 Certificate (can be determined by running OpenSSL's `openssl x509` command: `openssl x509 -noout -subject -in certificate.crt`). Note that no password is obtained from the user. Every entry in the user file needs this password: ``xj31ZMTZzkVA"`, 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|html)$">
SSLOptions +StdEnvVars +CompatEnvVars -ExportCertData
<Files>
```



Description:	Type of pass phrase dialog for encrypted private keys
Syntax:	SSLPassPhraseDialog <i>type</i>
Default:	SSLPassPhraseDialog builtin
Context:	server config
Status:	Extension
Module:	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).

- `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`" or `DSA`"), 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
```



Description:	Configure usable SSL protocol flavors
Syntax:	SSLProtocol [+ -] <i>protocol</i> ...
Default:	SSLProtocol all
Context:	server config, virtual host
Override:	Options
Status:	Extension
Module:	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.
- SSLv3
This is the Secure Sockets Layer (SSL) protocol, version 3.0. It is the successor to SSLv2 and the currently (as of February 1999) de-facto standardized SSL protocol from Netscape Corporation. 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 currently (as of February 1999) still under construction by the Internet Engineering Task Force (IETF). It's still not supported by any popular browsers.
- All
This is a shortcut for "+SSLv2 +SSLv3 +TLSv1" and a

convenient way for enabling all protocols except one when used in combination with the minus sign on a protocol as the example above shows.

Example

```
# enable SSLv3 and TLSv1, but not SSLv2  
SSLProtocol all -SSLv2
```



Description:	File of concatenated PEM-encoded CA Certificates for Remote Server Auth
Syntax:	SSLProxyCACertificateFile <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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
```



Description:	Directory of PEM-encoded CA Certificates for Remote Server Auth
Syntax:	SSLProxyCACertificatePath <i>directory-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

Example

```
SSLProxyCACertificatePath /usr/local/apache2/conf/ssl.crt/
```



Description:	File of concatenated PEM-encoded CA CRLs for Remote Server Auth
Syntax:	SSLProxyCAREvocationFile <i>file-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCAREvocationPath](#).

Example

```
SSLProxyCAREvocationFile /usr/local/apache2/conf/ssl.crl/ca-  
bundle-remote-server.crl
```



Description:	Directory of PEM-encoded CA CRLs for Remote Server Auth
Syntax:	SSLProxyCAREvocationPath <i>directory-path</i>
Context:	server config, virtual host
Status:	Extension
Module:	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. Use the `Makefile` which comes with [mod_ssl](#) to accomplish this task.

Example

```
SSLProxyCAREvocationPath /usr/local/apache2/conf/ssl.crl/
```



Description:	Cipher Suite available for negotiation in SSL proxy h
Syntax:	SSLProxyCipherSuite <i>cipher-spec</i>
Default:	SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
Context:	server config, virtual host, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_ssl

Equivalent to SSLCipherSuite, but for the proxy connection.
Please refer to [SSLCipherSuite](#) for additional information.



SSLProxyEngine Directive

Description:	SSL Proxy Engine Operation Switch
Syntax:	SSLProxyEngine on off
Default:	SSLProxyEngine off
Context:	server config, virtual host
Status:	Extension
Module:	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 image both for the main server and all configured virtual hosts.

Example

```
<VirtualHost _default_:443>  
SSLProxyEngine on  
...  
</VirtualHost>
```



Description:	File of concatenated PEM-encoded client certificates and keys to be used by the proxy
Syntax:	SSLProxyMachineCertificateFile <i>filename</i>
Context:	server config
Override:	Not applicable
Status:	Extension
Module:	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
```



Description:	Directory of PEM-encoded client certificates and keys to be used by the proxy
Syntax:	SSLProxyMachineCertificatePath <i>directory</i>
Context:	server config
Override:	Not applicable
Status:	Extension
Module:	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. Use the Makefile which comes with mod_ssl to accomplish this task.

Currently there is no support for encrypted private keys

Example:

```
SSLProxyMachineCertificatePath  
/usr/local/apache2/conf/proxy.crt/
```



Description:	Configure usable SSL protocol flavors for proxy usage
Syntax:	SSLProxyProtocol [+ -] <i>protocol</i> ...
Default:	SSLProxyProtocol all
Context:	server config, virtual host
Override:	Options
Status:	Extension
Module:	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.



Description:	Type of remote server Certificate verification
Syntax:	SSLProxyVerify <i>level</i>
Default:	SSLProxyVerify none
Context:	server config, virtual host, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_ssl

This directive sets the Certificate verification level for the remote server Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the remote server 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 remote server verification level after the HTTP request was read but before the HTTP response is sent.

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

Description:	Maximum depth of CA Certificates in Remote Server Certificate verification
Syntax:	SSLProxyVerifyDepth <i>number</i>
Default:	SSLProxyVerifyDepth 1
Context:	server config, virtual host, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_ssl

This directive sets how deeply mod_ssl should verify before deciding that the remote server does not 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 remote server 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 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
```



Description:	Pseudo Random Number Generator (PRNG) seeding source
Syntax:	SSLRandomSeed <i>context source [bytes]</i>
Context:	server config
Status:	Extension
Module:	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. It's 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 tool like [EGD](#) (Entropy Gathering Daemon) and run it's 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
```



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's a very powerful directive because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks.

The *expression* must match the following syntax (given as a BNF grammar notation):

```
expr ::= "true" | "false"
      | "!" expr
      | expr "&&" expr
      | expr "||" expr
      | "(" expr ")"
      | comp
```

```
comp ::= word "==" word | word "eq" word
      | word "!=" word | word "ne" word
      | word "<" word | word "lt" word
      | word "<=" word | word "le" word
      | word ">" word | word "gt" word
      | word ">=" word | word "ge" word
      | word "in" "{" wordlist "}"
      | word "=~" 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 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]+$/

```

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_CHAINn	
SSL_CLIENT_VERIFY	



Description:	Deny access when SSL is not used for the HTTP request
Syntax:	SSLRequireSSL
Context:	directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	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
```



Description: Type of the global/inter-process SSL Session Cache

Syntax: SSLSessionCache *type*

Default: SSLSessionCache none

Context: server config

Status: Extension

Module: 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 two storage *types* are currently supported:

- none
This is the default and just disables the global/inter-process Session Cache. There is no drawback in functionality, but a noticeable speed penalty can be observed.
- 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. The slight increase in I/O on the server results in a visible request speedup for your clients, so this type of storage is generally recommended.
- shm:/path/to/datafile[(size)]

This makes use of a high-performance hash table (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 storage type is not available on all platforms.

Examples

```
SSLSessionCache dbm:/usr/local/apache/logs/ssl_gcache_data
SSLSessionCache
shm:/usr/local/apache/logs/ssl_gcache_data(512000)
```



Description:	Number of seconds before an SSL session expires in the Session Cache
Syntax:	SSLSessionCacheTimeout <i>seconds</i>
Default:	SSLSessionCacheTimeout 300
Context:	server config, virtual host
Status:	Extension
Module:	mod_ssl

This directive sets the timeout in seconds for the information stored in the global/inter-process SSL Session Cache and the OpenSSL internal memory cache. 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
```



Description:	Variable name to determine user name
Syntax:	SSLUserName <i>varname</i>
Context:	server config, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_ssl
Compatibility:	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](#).

Example

```
SSLUserName SSL_CLIENT_S_DN_CN
```



Description:	Type of Client Certificate verification
Syntax:	SSLVerifyClient <i>level</i>
Default:	SSLVerifyClient none
Context:	server config, virtual host, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	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
```



Description:	Maximum depth of CA Certificates in Client Certificate verification
Syntax:	SSLVerifyDepth <i>number</i>
Default:	SSLVerifyDepth 1
Context:	server config, virtual host, directory, .htaccess
Override:	AuthConfig
Status:	Extension
Module:	mod_ssl

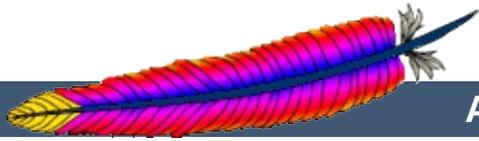
This directive sets how deeply mod_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

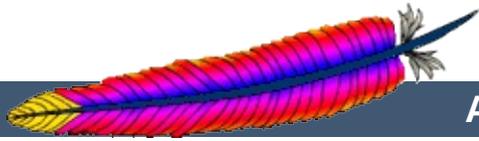
```
http://your.server.name/server-status?auto
    status . /support
log_server_status Perl .
```

```
mod_status ( , .htaccess
```



ExtendedStatus

```
ExtendedStatus On|Off
ExtendedStatus Off
Base
mod_status
ExtendedStatus 1.3.2
```



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

mod_suexec

- CGI
- Extension
- suexec_module
- mod_suexec.c
- 2.0

[suexec](#) CGI

[SuEXEC](#)

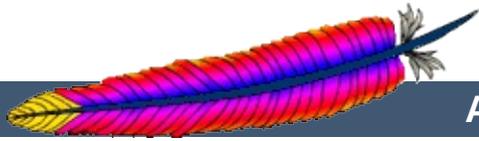


SuexecUserGroup

```
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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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 .
, .)

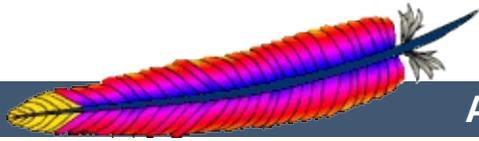
. ,
) . pid .

(

```

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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
```

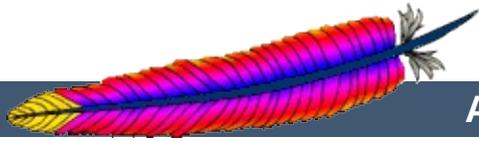
. :

```
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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module mod_usertrack

Description:	<i>Clickstream</i> logging of user activity on a site
Status:	Extension
Module Identifier:	usertrack_module
Source File:	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".



Description:	The domain to which the tracking cookie applies
Syntax:	CookieDomain <i>domain</i>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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, `.foo.com` is legal, but `foo.bar.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`).



Description:	Expiry time for the tracking cookie
Syntax:	CookieExpires <i>expiry-period</i>
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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.



Description:	Name of the tracking cookie
Syntax:	CookieName <i>token</i>
Default:	CookieName Apache
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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 "-".



Description:	Format of the cookie header field
Syntax:	CookieStyle <i>Netscape Cookie Cookie2 RFC2109 RFC2961</i>
Default:	CookieStyle Netscape
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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.



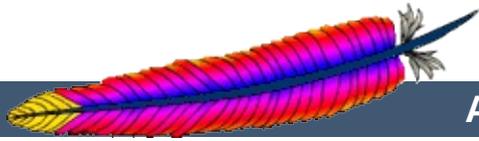
CookieTracking Directive

Description:	Enables tracking cookie
Syntax:	CookieTracking on off
Default:	CookieTracking off
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Extension
Module:	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 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

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

```

operator	
= ==	
>	
>=	
<	
<=	

```

<IfVersion >= 2.1>
# 2.1.0
# .
</IfVersion>

```

. .



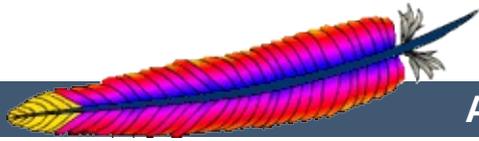
<i>operator</i>	
= ==	<i>version / regex/</i>
~	<i>version regex</i>

```
<IfVersion = /^2.1.[01234]$/>
# ,
</IfVersion>
```

(!) .

```
<IfVersion !~ ^2.1.[01234]$>
#
</IfVersion>
```

operator = .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

Apache Module `mod_vhost_alias`

Description:	Provides for dynamically configured mass virtual hosting
Status:	Extension
Module Identifier:	<code>vhost_alias_module</code>
Source File:	<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.



Description:	Dynamically configure the location of the document root for a given virtual host
Syntax:	<code>VirtualDocumentRoot</code> <i>interpolated-directory</i> none
Default:	<code>VirtualDocumentRoot</code> none
Context:	server config, virtual host
Status:	Extension
Module:	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`.



Description:	Dynamically configure the location of the document root for a given virtual host
Syntax:	<code>VirtualDocumentRootIP <i>interpolated-directory</i> none</code>
Default:	<code>VirtualDocumentRootIP none</code>
Context:	server config, virtual host
Status:	Extension
Module:	<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

Description:	Dynamically configure the location of the CGI directory for a given virtual host
Syntax:	<code>VirtualScriptAlias <i>interpolated-directory</i> none</code>
Default:	<code>VirtualScriptAlias none</code>
Context:	server config, virtual host
Status:	Extension
Module:	<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.

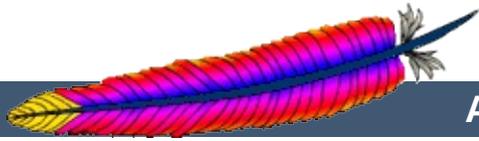


Description:	Dynamically configure the location of the cgi directory for a given virtual host
Syntax:	<code>VirtualScriptAliasIP <i>interpolated-directory</i> none</code>
Default:	<code>VirtualScriptAliasIP none</code>
Context:	server config, virtual host
Status:	Extension
Module:	<code>mod_vhost_alias</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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 `srmd.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).

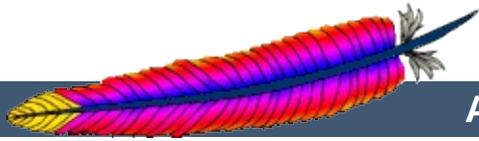
The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the `alias` module, which also indicates how a syntax error can be returned (note that the per-directory configuration 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

Debugging Memory Allocation in APR

The allocation mechanism's within APR have a number of debugging modes that can be used to assist in finding memory problems. This document describes the modes available and gives instructions on activating them.



Allocation Debugging - `ALLOC_DEBUG`

Debugging support: Define this to enable code which helps detect re-use of `free()`d memory and other such nonsense.

The theory is simple. The `FILL_BYTE` (`0xa5`) is written over all `malloc`'d memory as we receive it, and is written over everything that we free up during a `clear_pool`. We check that blocks on the free list always have the `FILL_BYTE` in them, and we check during `palloc()` that the bytes still have `FILL_BYTE` in them. If you ever see garbage URLs or whatnot containing lots of `0xa5`s then you know something used data that's been freed or uninitialized.

Malloc Support - `ALLOC_USE_MALLOC`

If defined all allocations will be done with `malloc()` and `free()`d appropriately at the end.

This is intended to be used with something like Electric Fence or Purify to help detect memory problems. Note that if you're using `efence` then you should also add in `ALLOC_DEBUG`. But don't add in `ALLOC_DEBUG` if you're using Purify because `ALLOC_DEBUG` would hide all the uninitialized read errors that Purify can diagnose.

Pool Debugging - `POOL_DEBUG`

This is intended to detect cases where the wrong pool is used when assigning data to an object in another pool.

In particular, it causes the `table_{set, add, merge}` routines to check that their arguments are safe for the `apr_table_t` they're being placed in. It currently only works with the unix multiprocess model, but could be extended to others.

Table Debugging - MAKE_TABLE_PROFILE

Provide diagnostic information about `make_table()` calls which are possibly too small.

This requires a recent gcc which supports `__builtin_return_address()`. The `error_log` output will be a message such as:

```
table_push: apr_table_t created by 0x804d874 hit limit of 10
```

Use `l *0x804d874` to find the source that corresponds to. It indicates that a `apr_table_t` allocated by a call at that address has possibly too small an initial `apr_table_t` size guess.

Allocation Statistics - ALLOC_STATS

Provide some statistics on the cost of allocations.

This requires a bit of an understanding of how `alloc.c` works.



Not all the options outlined above can be activated at the same time. the following table gives more information.

	ALLOC DEBUG	ALLOC USE MALLOC	POOL DEBUG	MAKE TABLE PROFILE	ALLOC STATS
ALLOC DEBUG	-	No	Yes	Yes	Yes
ALLOC USE MALLOC	No	-	No	No	No
POOL DEBUG	Yes	No	-	Yes	Yes
MAKE TABLE PROFILE	Yes	No	Yes	-	Yes
ALLOC STATS	Yes	No	Yes	Yes	-

Additionally the debugging options are not suitable for multi-threaded versions of the server. When trying to debug with these options the server should be started in single process mode.



Remaining Debugging Options

The various options for debugging memory are now enabled in the `apr_general.h` header file in APR. The various options are enabled by uncommenting the define for the option you wish to use. The section of the code currently looks like this (*contained in `src/lib/apr/include/apr_pools.h`*)

```
/*
#define ALLOC_DEBUG
#define POOL_DEBUG
#define ALLOC_USE_MALLOC
#define MAKE_TABLE_PROFILE
#define ALLOC_STATS
*/

typedef struct ap_pool_t {
    union block_hdr *first;
    union block_hdr *last;
    struct cleanup *cleanups;
    struct process_chain *subprocesses;
    struct ap_pool_t *sub_pools;
    struct ap_pool_t *sub_next;
    struct ap_pool_t *sub_prev;
    struct ap_pool_t *parent;
    char *free_first_avail;
#ifdef ALLOC_USE_MALLOC
    void *allocation_list;
#endif
#ifdef POOL_DEBUG
    struct ap_pool_t *joined;
#endif
    int (*apr_abort)(int retcode);
    struct datastruct *prog_data;
} ap_pool_t;
```

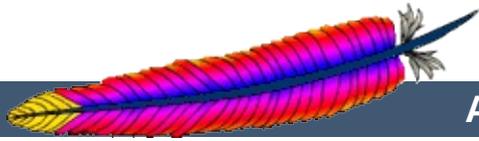
To enable allocation debugging simply move the `#define ALLOC_DEBUG` above the start of the comments block and rebuild the server.

Note

In order to use the various options the server **must** be rebuilt after editing the header file.

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 `>` 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:

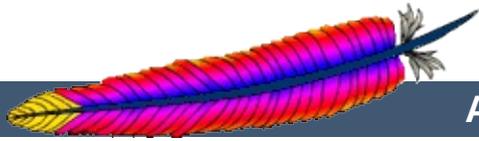
```
/**
 * @package Name of library header
 */
```

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 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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 tradition Apache hook return 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, Apache 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,
        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: HOOK_FIRST, HOOK_MIDDLE and HOOK_LAST.

All modules using any particular value may be run in any order relative to each other, but, of course, all modules using HOOK_FIRST will be run before HOOK_MIDDLE which are before HOOK_LAST. Modules that don't care when they are run should use HOOK_MIDDLE. (*I spaced these out so people could do stuff like HOOK_FIRST-2 to get in slightly earlier, but is this wise? - Ben*)

Note that there are two more values, HOOK_REALLY_FIRST and 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:

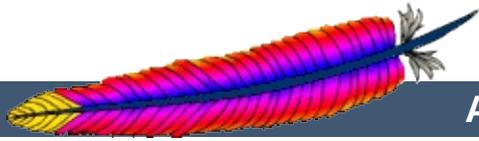
```
static void register_hooks()
{
    static const char * const aszPre[] = { "mod_xyz.c",
        "mod_abc.c", NULL };

    ap_hook_do_something(my_something_doer, aszPre, NULL,
        HOOK_MIDDLE);
}
```

Note that the sort used to achieve this is stable, so ordering set by HOOK_ORDER is preserved, as far as is possible.

Ben Laurie, 15th August 1999

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



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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

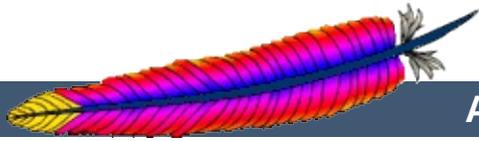
ap_hook_type_checker

determine and/or set the doc type

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

Request Processing in Apache 2.0

Warning

Warning - this is a first (fast) draft that needs further revision!

Several changes in Apache 2.0 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 Apache 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 Apache 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 Apache 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

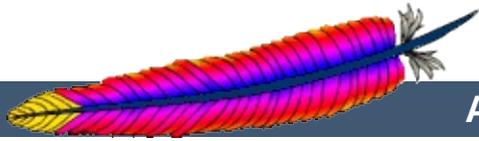
This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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.

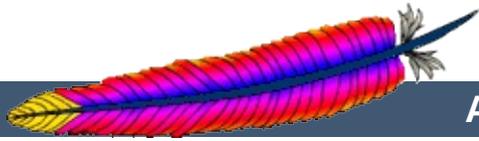
This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.



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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



• • • • • (; , , , , , .)



(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](#)

: [\(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 Prot
(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/2.0/glossary.html

.

Uniform Resource Identifier (URI)

. [RFC 2396](#) . URI

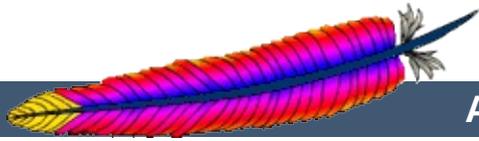
[URL](#) .

(Virtual Hosting)

. IP IP . (name-based)
IP .
:

X.509

(International Telecommunication Union, ITU-T) .
SSL/TLS .
: [SSL/TLS](#)



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >

_ .

A | B | C | D | E | F | G | H | I | K | L | M | N | O | P |
R | S | T | U | V | W | X

- [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_Authoritative](#)
- [Anonymous_LogEmail](#)
- [Anonymous_MustGiveEmail](#)
- [Anonymous_NoUserID](#)
- [Anonymous_VerifyEmail](#)
- [AssignUserID](#)
- [AuthAuthoritative](#)
- [AuthDBMAuthoritative](#)
- [AuthDBMGroupFile](#)
- [AuthDBMType](#)
- [AuthDBMUserFile](#)
- [AuthDigestAlgorithm](#)
- [AuthDigestDomain](#)
- [AuthDigestFile](#)
- [AuthDigestGroupFile](#)
- [AuthDigestNcCheck](#)
- [AuthDigestNonceFormat](#)
- [AuthDigestNonceLifetime](#)
- [AuthDigestQop](#)
- [AuthDigestShmemSize](#)
- [AuthGroupFile](#)
- [AuthLDAPAuthoritative](#)
- [AuthLDAPBindDN](#)
- [AuthLDAPBindPassword](#)
- [AuthLDAPCharsetConfig](#)
- [AuthLDAPCompareDNOnServer](#)
- [AuthLDAPDereferenceAliases](#)
- [AuthLDAPEnabled](#)
- [AuthLDAPFrontPageHack](#)
- [AuthLDAPGroupAttribute](#)
- [AuthLDAPGroupAttributeIsDN](#)

- [AuthLDAPRemoteUserIsDN](#)
- [AuthLDAPUrl](#)
- [AuthName](#)
- [AuthType](#)
- [AuthUserFile](#)
- [BrowserMatch](#)
- [BrowserMatchNoCase](#)
- [BS2000Account](#)
- [BufferedLogs](#)
- [CacheDefaultExpire](#)
- [CacheDirLength](#)
- [CacheDirLevels](#)
- [CacheDisable](#)
- [CacheEnable](#)
- [CacheExpiryCheck](#)
- [CacheFile](#)
- [CacheForceCompletion](#)
- [CacheGcClean](#)
- [CacheGcDaily](#)
- [CacheGcInterval](#)
- [CacheGcMemUsage](#)
- [CacheGcUnused](#)
- [CacheIgnoreCacheControl](#)
- [CacheIgnoreHeaders](#)
- [CacheIgnoreNoLastMod](#)
- [CacheLastModifiedFactor](#)
- [CacheMaxExpire](#)
- [CacheMaxFileSize](#)
- [CacheMinFileSize](#)
- [CacheNegotiatedDocs](#)
- [CacheRoot](#)
- [CacheSize](#)
- [CacheTimeMargin](#)
- [CGIMapExtension](#)

- [CharsetDefault](#)
- [CharsetOptions](#)
- [CharsetSourceEnc](#)
- [CheckSpelling](#)
- [ChildPerUserID](#)
- [ContentDigest](#)
- [CookieDomain](#)
- [CookieExpires](#)
- [CookieLog](#)
- [CookieName](#)
- [CookieStyle](#)
- [CookieTracking](#)
- [CoreDumpDirectory](#)
- [CustomLog](#)
- [Dav](#)
- [DavDepthInfinity](#)
- [DavLockDB](#)
- [DavMinTimeout](#)
- [DefaultIcon](#)
- [DefaultLanguage](#)
- [DefaultType](#)
- [DeflateBufferSize](#)
- [DeflateCompressionLevel](#)
- [DeflateFilterNote](#)
- [DeflateMemLevel](#)
- [DeflateWindowSize](#)
- [Deny](#)
- [<Directory>](#)
- [DirectoryIndex](#)
- [<DirectoryMatch>](#)
- [DirectorySlash](#)
- [DocumentRoot](#)
- [DumpIOInput](#)
- [DumpIOOutput](#)

- [EnableExceptionHook](#)
- [EnableMMAP](#)
- [EnableSendfile](#)
- [ErrorDocument](#)
- [ErrorLog](#)
- [Example](#)
- [ExpiresActive](#)
- [ExpiresByType](#)
- [ExpiresDefault](#)
- [ExtendedStatus](#)
- [ExtFilterDefine](#)
- [ExtFilterOptions](#)
- [FileETag](#)
- [<Files>](#)
- [<FilesMatch>](#)
- [ForceLanguagePriority](#)
- [ForceType](#)
- [ForensicLog](#)
- [Group](#)
- [Header](#)
- [HeaderName](#)
- [HostnameLookups](#)
- [IdentityCheck](#)
- [<IfDefine>](#)
- [<IfModule>](#)
- [<IfVersion>](#)
- [ImapBase](#)
- [ImapDefault](#)
- [ImapMenu](#)
- [Include](#)
- [IndexIgnore](#)
- [IndexOptions](#)
- [IndexOrderDefault](#)
- [ISAPIAppendLogToErrors](#)

- [ISAPIAppendLogToQuery](#)
- [ISAPICacheFile](#)
- [ISAPIFakeAsync](#)
- [ISAPILogNotSupported](#)
- [ISAPIReadAheadBuffer](#)
- [KeepAlive](#)
- [KeepAliveTimeout](#)
- [LanguagePriority](#)
- [LDAPCacheEntries](#)
- [LDAPCacheTTL](#)
- [LDAPConnectionTimeout](#)
- [LDAPOpCacheEntries](#)
- [LDAPOpCacheTTL](#)
- [LDAPSharedCacheFile](#)
- [LDAPSharedCacheSize](#)
- [LDAPTrustedCA](#)
- [LDAPTrustedCAType](#)
- [<Limit>](#)
- [<LimitExcept>](#)
- [LimitInternalRecursion](#)
- [LimitRequestBody](#)
- [LimitRequestFields](#)
- [LimitRequestFieldSize](#)
- [LimitRequestLine](#)
- [LimitXMLRequestBody](#)
- [Listen](#)
- [ListenBackLog](#)
- [LoadFile](#)
- [LoadModule](#)
- [<Location>](#)
- [<LocationMatch>](#)
- [LockFile](#)
- [LogFormat](#)
- [LogLevel](#)

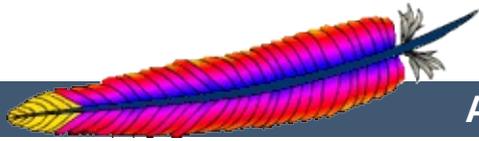
- [MaxClients](#)
- [MaxKeepAliveRequests](#)
- [MaxMemFree](#)
- [MaxRanges](#)
- [MaxRequestsPerChild](#)
- [MaxRequestsPerThread](#)
- [MaxSpareServers](#)
- [MaxSpareThreads](#)
- [MaxThreads](#)
- [MaxThreadsPerChild](#)
- [MCacheMaxObjectCount](#)
- [MCacheMaxObjectSize](#)
- [MCacheMaxStreamingBuffer](#)
- [MCacheMinObjectSize](#)
- [MCacheRemovalAlgorithm](#)
- [MCacheSize](#)
- [MetaDir](#)
- [MetaFiles](#)
- [MetaSuffix](#)
- [MimeMagicFile](#)
- [MinSpareServers](#)
- [MinSpareThreads](#)
- [MMapFile](#)
- [ModMimeUsePathInfo](#)
- [MultiviewsMatch](#)
- [NameVirtualHost](#)
- [NoProxy](#)
- [NumServers](#)
- [NWSSLTrustedCerts](#)
- [NWSSLUpgradeable](#)
- [Options](#)
- [Order](#)
- [PassEnv](#)
- [PidFile](#)

- [ProtocolEcho](#)
- [<Proxy>](#)
- [ProxyBadHeader](#)
- [ProxyBlock](#)
- [ProxyDomain](#)
- [ProxyErrorOverride](#)
- [ProxyFtpDirCharset](#)
- [ProxyIOBufferSize](#)
- [<ProxyMatch>](#)
- [ProxyMaxForwards](#)
- [ProxyPass](#)
- [ProxyPassReverse](#)
- [ProxyPreserveHost](#)
- [ProxyReceiveBufferSize](#)
- [ProxyRemote](#)
- [ProxyRemoteMatch](#)
- [ProxyRequests](#)
- [ProxyTimeout](#)
- [ProxyVia](#)
- [ReadmeName](#)
- [ReceiveBufferSize](#)
- [Redirect](#)
- [RedirectMatch](#)
- [RedirectPermanent](#)
- [RedirectTemp](#)
- [RemoveCharset](#)
- [RemoveEncoding](#)
- [RemoveHandler](#)
- [RemoveInputFilter](#)
- [RemoveLanguage](#)
- [RemoveOutputFilter](#)
- [RemoveType](#)
- [RequestHeader](#)
- [Require](#)

- [RewriteBase](#)
- [RewriteCond](#)
- [RewriteEngine](#)
- [RewriteLock](#)
- [RewriteLog](#)
- [RewriteLogLevel](#)
- [RewriteMap](#)
- [RewriteOptions](#)
- [RewriteRule](#)
- [RLimitCPU](#)
- [RLimitMEM](#)
- [RLimitNPROC](#)
- [Satisfy](#)
- [ScoreBoardFile](#)
- [Script](#)
- [ScriptAlias](#)
- [ScriptAliasMatch](#)
- [ScriptInterpreterSource](#)
- [ScriptLog](#)
- [ScriptLogBuffer](#)
- [ScriptLogLength](#)
- [ScriptSock](#)
- [SecureListen](#)
- [SendBufferSize](#)
- [ServerAdmin](#)
- [ServerAlias](#)
- [ServerLimit](#)
- [ServerName](#)
- [ServerPath](#)
- [ServerRoot](#)
- [ServerSignature](#)
- [ServerTokens](#)
- [SetEnv](#)
- [SetEnvIf](#)

- [SetEnvIfNoCase](#)
- [SetHandler](#)
- [SetInputFilter](#)
- [SetOutputFilter](#)
- [SSIEndTag](#)
- [SSIErrorMsg](#)
- [SSIStartTag](#)
- [SSITimeFormat](#)
- [SSIUndefinedEcho](#)
- [SSLCACertificateFile](#)
- [SSLCACertificatePath](#)
- [SSLCARevocationFile](#)
- [SSLCARevocationPath](#)
- [SSLCertificateChainFile](#)
- [SSLCertificateFile](#)
- [SSLCertificateKeyFile](#)
- [SSLCipherSuite](#)
- [SSLEngine](#)
- [SSLHonorCipherOrder](#)
- [SSLInsecureRenegotiation](#)
- [SSLMutex](#)
- [SSLOptions](#)
- [SSLPassPhraseDialog](#)
- [SSLProtocol](#)
- [SSLProxyCACertificateFile](#)
- [SSLProxyCACertificatePath](#)
- [SSLProxyCARevocationFile](#)
- [SSLProxyCARevocationPath](#)
- [SSLProxyCipherSuite](#)
- [SSLProxyEngine](#)
- [SSLProxyMachineCertificateFile](#)
- [SSLProxyMachineCertificatePath](#)
- [SSLProxyProtocol](#)
- [SSLProxyVerify](#)

- [SSLProxyVerifyDepth](#)
- [SSLRandomSeed](#)
- [SSLRequire](#)
- [SSLRequireSSL](#)
- [SSLSessionCache](#)
- [SSLSessionCacheTimeout](#)
- [SSLUserName](#)
- [SSLVerifyClient](#)
- [SSLVerifyDepth](#)
- [StartServers](#)
- [StartThreads](#)
- [SuexecUserGroup](#)
- [ThreadLimit](#)
- [ThreadsPerChild](#)
- [ThreadStackSize](#)
- [TimeOut](#)
- [TraceEnable](#)
- [TransferLog](#)
- [TypesConfig](#)
- [UnsetEnv](#)
- [UseCanonicalName](#)
- [User](#)
- [UserDir](#)
- [VirtualDocumentRoot](#)
- [VirtualDocumentRootIP](#)
- [<VirtualHost>](#)
- [VirtualScriptAlias](#)
- [VirtualScriptAliasIP](#)
- [Win32DisableAcceptEx](#)
- [XBitHack](#)



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) >



... ▢

.

.

A | **B** | **C** | **D** | **E** | **F** | **G** | **H** |
I | **K** | **L** | **M** | **N** | **O** | **P** | **R** |
S | **T** | **U** | **V** | **W** | **X**

s	C Core
v	M MPM
d directory	B Base
h .htaccess	E Extension
	X Experimental

AcceptMutex <i>Default</i> Method that Apache uses to serialize multiple children accepting requests on network sockets	Default
AcceptPathInfo <i>On Off Default</i> Resources accept trailing pathname information	Default
AccessFileName <i>filename [filename] ...</i> Name of the distributed configuration file	.htaccess
Action <i>action-type cgi-script</i> content-type CGI	
AddAlt <i>string file [file] ...</i>	
AddAltByEncoding <i>string MIME-encoding [MIME-encoding] ...</i> MIME-encoding	
AddAltByType <i>string MIME-type [MIME-type] ...</i> MIME content-type	
AddCharset <i>charset extension [extension] ...</i>	

<p>Maps the given filename extensions to the specified content charset</p> <p>AddDefaultCharset <i>On Off charset</i></p>	Off
<p>Default charset parameter to be added when a response content-type is text/plain or text/html</p>	
<p>AddDescription <i>string file [file] ...</i></p>	
<p>AddEncoding <i>MIME-enc extension [extension]</i></p> <p>...</p> <p>Maps the given filename extensions to the specified encoding type</p>	
<p>AddHandler <i>handler-name extension [extension] ...</i></p> <p>Maps the filename extensions to the specified handler</p>	
<p>AddIcon <i>icon name [name] ...</i></p>	
<p>AddIconByEncoding <i>icon MIME-encoding [MIME-encoding] ...</i></p> <p>MIME content-encoding</p>	
<p>AddIconByType <i>icon MIME-type [MIME-type]</i></p> <p>...</p> <p>MIME content-type</p>	
<p>AddInputFilter <i>filter[:filter...] extension [extension] ...</i></p> <p>Maps filename extensions to the filters that will process client requests</p>	
<p>AddLanguage <i>MIME-lang extension [extension] ...</i></p> <p>Maps the given filename extension to the specified content language</p>	
<p>AddModuleInfo <i>module-name string</i></p> <p>server-info</p>	
<p>AddOutputFilter <i>filter[:filter...] extension [extension] ...</i></p> <p>Maps filename extensions to the filters that will process responses from the server</p>	
<p>AddOutputFilterByType <i>filter[:filter...] MIME-type [MIME-type] ...</i></p> <p>assigns an output filter to a particular MIME-type</p>	
<p>AddType <i>MIME-type extension [extension] ...</i></p> <p>Maps the given filename extensions onto the specified content type</p>	

Alias <i>URL-path file-path directory-path</i>	
URL	
AliasMatch <i>regex file-path directory-path</i>	
URL	
Allow from <i>all host env=env-variable</i> [host env=env-variable] ...	
Controls which hosts can access an area of the server	
AllowCONNECT <i>port [port] ...</i>	443 563
Ports that are allowed to CONNECT through the proxy	
AllowEncodedSlashes <i>On Off</i>	Off
Determines whether encoded path separators in URLs are allowed to be passed through	
AllowOverride <i>All None directive-type</i> [directive-type] ...	All
Types of directives that are allowed in .htaccess files	
Anonymous <i>user [user] ...</i>	
Specifies userIDs that are allowed access without password verification	
Anonymous_Authoritative <i>On Off</i>	Off
Configures if authorization will fall-through to other methods	
Anonymous_LogEmail <i>On Off</i>	On
Sets whether the password entered will be logged in the error log	
Anonymous_MustGiveEmail <i>On Off</i>	On
Specifies whether blank passwords are allowed	
Anonymous_NoUserID <i>On Off</i>	Off
Sets whether the userID field may be empty	
Anonymous_VerifyEmail <i>On Off</i>	Off
Sets whether to check the password field for a correctly formatted email address	
AssignUserID <i>user-id group-id</i>	
Tie a virtual host to a user and group ID	
AuthAuthoritative <i>On Off</i>	On
Sets whether authorization and authentication are passed to lower level modules	
AuthDBMAuthoritative <i>On Off</i>	On
Sets whether authentication and authorization will be passed on to lower level modules	
AuthDBMGroupFile <i>file-path</i>	
Sets the name of the database file containing the list of user groups for authentication	
AuthDBMType	default

default SDBM GDBM NDBM DB	
Sets the type of database file that is used to store passwords	
AuthDBMUserFile <i>file-path</i>	
Sets the name of a database file containing the list of users and passwords for authentication	
AuthDigestAlgorithm MD5 MD5-sess	MD5
digest authentication challenge response hash	
AuthDigestDomain <i>URI [URI] ...</i>	
digest authentication URI	
AuthDigestFile <i>file-path</i>	
digest authentication	
AuthDigestGroupFile <i>file-path</i>	
digest authentication	
AuthDigestNcCheck On Off	Off
nonce-count	
AuthDigestNonceFormat <i>format</i>	
nonce	
AuthDigestNonceLifetime <i>seconds</i>	300
nonce	
AuthDigestQop none auth auth-int [auth auth-int]	auth
digest authentication (quality-of-protection) .	
AuthDigestShmemSize <i>size</i>	1000
AuthGroupFile <i>file-path</i>	
Sets the name of a text file containing the list of user groups for authentication	
AuthLDAPAuthoritative on off	on
Prevent other authentication modules from authenticating the user if this one fails	
AuthLDAPBindDN <i>distinguished-name</i>	
Optional DN to use in binding to the LDAP server	
AuthLDAPBindPassword <i>password</i>	
Password used in conjunction with the bind DN	
AuthLDAPCharsetConfig <i>file-path</i>	
Language to charset conversion configuration file	
AuthLDAPCompareDNOnServer on off	on
Use the LDAP server to compare the DNS	

AuthLDAPDereferenceAliases never searching finding always	Always
When will the module de-reference aliases	
AuthLDAPEnabled on off	on
Turn on or off LDAP authentication	
AuthLDAPFrontPageHack on off	off
Allow LDAP authentication to work with MS FrontPage	
AuthLDAPGroupAttribute <i>attribute</i>	
LDAP attributes used to check for group membership	
AuthLDAPGroupAttributeIsDN on off	on
Use the DN of the client username when checking for group membership	
AuthLDAPRemoteUserIsDN on off	off
Use the DN of the client username to set the REMOTE_USER environment variable	
AuthLDAPUrl <i>url</i>	
URL specifying the LDAP search parameters	
AuthName <i>auth-domain</i>	
Authorization realm for use in HTTP authentication	
AuthType Basic Digest	
Type of user authentication	
AuthUserFile <i>file-path</i>	
Sets the name of a text file containing the list of users and passwords for authentication	
BrowserMatch <i>regex [!]env-variable[=value] [!]env-variable[=value] ...</i>	
HTTP User-Agent	
BrowserMatchNoCase <i>regex [!]env-variable[=value] [!]env-variable[=value] ...</i>	
User-Agent	
BS2000Account <i>account</i>	
Define the non-privileged account on BS2000 machines	
BufferedLogs On Off	Off
Buffer log entries in memory before writing to disk	
CacheDefaultExpire <i>seconds</i>	3600 (one hour)
.	
CacheDirLength <i>length</i>	2

CacheDirLevels <i>levels</i>	3
CacheDisable <i>url-string</i> URL	
CacheEnable <i>cache type url-string</i> URL	
CacheExpiryCheck On Off	On
CacheFile <i>file-path [file-path] ...</i>	
CacheForceCompletion <i>Percentage</i>	60
CacheGcClean <i>hours url-string</i> URL	?
CacheGcDaily <i>time</i> (24)	?
CacheGcInterval <i>hours</i>	
CacheGcMemUsage <i>KBytes</i> (kilobyte)	?
CacheGcUnused <i>hours url-string</i> URL	?
CacheIgnoreCacheControl On Off	Off
CacheIgnoreHeaders <i>header-string [header-string] ...</i> Do not store the given HTTP header(s) in the cache.	None
CacheIgnoreNoLastMod On Off Last Modified	Off
CacheLastModifiedFactor <i>float</i> LastModified	0.1
CacheMaxExpire <i>seconds</i>	86400 ()
CacheMaxFileSize <i>bytes</i> ()	1000000

CacheMinFileSize <i>bytes</i> ()	1
CacheNegotiatedDocs On Off Allows content-negotiated documents to be cached by proxy servers	Off
CacheRoot <i>directory</i> root	
CacheSize <i>KBytes</i> (KByte)	1000000
CacheTimeMargin ?	?
CGIMapExtension <i>cgi-path .extension</i> Technique for locating the interpreter for CGI scripts	
CharsetDefault <i>charset</i>	
CharsetOptions <i>option [option] ...</i>	DebugLevel=0 NoImp
CharsetSourceEnc <i>charset</i>	
CheckSpelling on off	Off
ChildPerUserID <i>user-id group-id num-children</i> Specify user ID and group ID for a number of child processes	
ContentDigest On Off Enables the generation of Content -MD5 HTTP Response headers	Off
CookieDomain <i>domain</i> The domain to which the tracking cookie applies	
CookieExpires <i>expiry-period</i> Expiry time for the tracking cookie	
CookieLog <i>filename</i>	
CookieName <i>token</i> Name of the tracking cookie	Apache
CookieStyle Netscape Cookie Cookie2 RFC2109 RFC2965 Format of the cookie header field	Netscape

CookieTracking on off	off
Enables tracking cookie	
CoreDumpDirectory <i>directory</i>	
Directory where Apache attempts to switch before dumping core	
CustomLog <i>file pipe format nickname [env=</i> [!]<i>environment-variable</i>	
Dav On Off <i>provider-name</i>	Off
WebDAV HTTP	
DavDepthInfinity on off	off
PROPFIND Depth: Infinity	
DavLockDB <i>file-path</i>	
DAV	
DavMinTimeout <i>seconds</i>	0
DAV	
DefaultIcon <i>url-path</i>	
DefaultLanguage <i>MIME-lang</i>	
Sets all files in the given scope to the specified language	
DefaultType <i>MIME-type</i>	text/plain
MIME content-type that will be sent if the server cannot determine a type in any other way	
DeflateBufferSize <i>value</i>	8096
zlib	
DeflateCompressionLevel <i>value</i>	
DeflateFilterNote [<i>type</i>] <i>notename</i>	
DeflateMemLevel <i>value</i>	9
zlib	
DeflateWindowSize <i>value</i>	15
Zlib window size	
Deny from all <i>host</i> <i>env=env-variable</i> [<i>host</i> <i>env=env-variable</i>] ...	
Controls which hosts are denied access to the server	
<Directory <i>directory-path</i>> ... </Directory>	

Enclose a group of directives that apply only to the named file-system directory and sub-directories	
DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...	index.html
<DirectoryMatch <i>regex</i> ... </DirectoryMatch>	
Enclose directives that apply to file-system directories matching a regular expression and their subdirectories	
DirectorySlash On Off	On
DocumentRoot <i>directory-path</i>	/usr/local/apache/h +
Directory that forms the main document tree visible from the web	
DumpIOInput On Off	Off
Dump all input data to the error log	
DumpIOOutput On Off	Off
Dump all output data to the error log	
EnableExceptionHook On Off	Off
Enables a hook that runs exception handlers after a crash	
EnableMMAP On Off	On
Use memory-mapping to read files during delivery	
EnableSendfile On Off	On
Use the kernel sendfile support to deliver files to the client	
ErrorDocument <i>error-code</i> <i>document</i>	
What the server will return to the client in case of an error	
ErrorLog <i>file-path</i> syslog[: <i>facility</i>]	logs/error_log (Uni +
Location where the server will log errors	
Example	
API	
ExpiresActive On Off	
Expires	
ExpiresByType <i>MIME-type</i> <code><code>seconds</code>	
MIME type Expires	
ExpiresDefault <code><code>seconds</code>	
ExtendedStatus On Off	Off
ExtFilterDefine <i>filtername</i> <i>parameters</i>	

ExtFilterOptions <i>option</i> [<i>option</i>] ...	DebugLevel=0 NoLog +
mod_ext_filter	
FileETag <i>component</i> ...	Inode MTime Size
File attributes used to create the ETag HTTP response header	
<Files <i>filename</i>> ... </Files>	
Contains directives that apply to matched filenames	
<FilesMatch <i>regex</i>> ... </FilesMatch>	
Contains directives that apply to regular-expression matched filenames	
ForceLanguagePriority None Prefer Fallback [Prefer Fallback]	Prefer
Action to take if a single acceptable document is not found	
ForceType <i>MIME-type</i> None	
Forces all matching files to be served with the specified MIME content-type	
ForensicLog <i>filename</i> <i>pipe</i>	
Sets filename of the forensic log	
Group <i>unix-group</i>	#-1
Group under which the server will answer requests	
Header [<i>condition</i>] set append add unset echo <i>header</i> [<i>value</i>] [<i>env</i>=[!]<i>variable</i>]	
HTTP	
HeaderName <i>filename</i>	
HostnameLookups On Off Double	Off
Enables DNS lookups on client IP addresses	
IdentityCheck On Off	Off
Enables logging of the RFC1413 identity of the remote user	
<IfDefine [!]<i>parameter-name</i>> ... </IfDefine>	
Encloses directives that will be processed only if a test is true at startup	
<IfModule [!]<i>module-name</i>> ... </IfModule>	
Encloses directives that are processed conditional on the presence or absence of a specific	
<IfVersion [[!]<i>operator</i>] <i>version</i>> ... </IfVersion>	

ImapBase map referer URL base	http://servername/
ImapDefault error nocontent map referer URL	nocontent
ImapMenu none formatted semiformatted unformatted	
Include file-path directory-path Includes other configuration files from within the server configuration files	
IndexIgnore file [file] ...	
IndexOptions [+ -]option [[+ -]option] ...	
IndexOrderDefault Ascending Descending Name Date Size Description	Ascending Name
ISAPIAppendLogToErrors on off ISAPI exntension HSE_APPEND_LOG_PARAMETER	off
ISAPIAppendLogToQuery on off ISAPI exntension HSE_APPEND_LOG_PARAMETER	on
ISAPICacheFile file-path [file-path] ... ISAPI .dll	
ISAPIFakeAsync on off ISAPI	off
ISAPILogNotSupported on off ISAPI extension	off
ISAPIReadAheadBuffer size ISAPI extension (read ahead buffer)	49152
KeepAlive On Off Enables HTTP persistent connections	On
KeepAliveTimeout seconds Amount of time the server will wait for subsequent requests on a persistent connection	15
LanguagePriority MIME-lang [MIME-lang] ... The precedence of language variants for cases where the client does not express a prefer	
LDAPCacheEntries number	1024

Maximum number of entries in the primary LDAP cache	
LDAPCacheTTL seconds	600
Time that cached items remain valid	
LDAPConnectionTimeout seconds	
Specifies the socket connection timeout in seconds	
LDAPOpCacheEntries number	1024
Number of entries used to cache LDAP compare operations	
LDAPOpCacheTTL seconds	600
Time that entries in the operation cache remain valid	
LDAPSharedCacheFile directory-path/filename	
Sets the shared memory cache file	
LDAPSharedCacheSize bytes	102400
Size in bytes of the shared-memory cache	
LDAPTrustedCA directory-path/filename	
Sets the file containing the trusted Certificate Authority certificate or database	
LDAPTrustedCAType type	
Specifies the type of the Certificate Authority file	
<Limit method [method] ... > ... </Limit>	
Restrict enclosed access controls to only certain HTTP methods	
<LimitExcept method [method] ... > ... </LimitExcept>	
Restrict access controls to all HTTP methods except the named ones	
LimitInternalRecursion number [number]	10
Determine maximum number of internal redirects and nested subrequests	
LimitRequestBody bytes	0
Restricts the total size of the HTTP request body sent from the client	
LimitRequestFields number	100
Limits the number of HTTP request header fields that will be accepted from the client	
LimitRequestFieldsize bytes	
Limits the size of the HTTP request header allowed from the client	
LimitRequestLine bytes	8190
Limit the size of the HTTP request line that will be accepted from the client	
LimitXMLRequestBody bytes	1000000
Limits the size of an XML-based request body	

<u>Listen</u> <i>[IP-address:]portnumber</i>	
IP addresses and ports that the server listens to	
<u>ListenBacklog</u> <i>backlog</i>	
Maximum length of the queue of pending connections	
<u>LoadFile</u> <i>filename [filename] ...</i>	
<u>LoadModule</u> <i>module filename</i>	
<u><Location URL-path URL> ... </Location></u>	
Applies the enclosed directives only to matching URLs	
<u><LocationMatch regex> ... </LocationMatch></u>	
Applies the enclosed directives only to regular-expression matching URLs	
<u>LockFile</u> <i>filename</i>	logs/accept.lock
Location of the accept serialization lock file	
<u>LogFormat</u> <i>format nickname [nickname]</i>	"%h %l %u %t \"%r\" -
<u>LogLevel</u> <i>level</i>	warn
Controls the verbosity of the ErrorLog	
<u>MaxClients</u> <i>number</i>	
Maximum number of connections that will be processed simultaneously	
<u>MaxKeepAliveRequests</u> <i>number</i>	100
Number of requests allowed on a persistent connection	
<u>MaxMemFree</u> <i>KBytes</i>	0
Maximum amount of memory that the main allocator is allowed to hold without calling free	
<u>MaxRanges</u> <i>default unlimited none number-of-ranges</i>	200
Number of ranges allowed before returning the complete resource	
<u>MaxRequestsPerChild</u> <i>number</i>	10000
Limit on the number of requests that an individual child server will handle during its life	
<u>MaxRequestsPerThread</u> <i>number</i>	0
<u>MaxSpareServers</u> <i>number</i>	10
Maximum number of idle child server processes	
<u>MaxSpareThreads</u> <i>number</i>	
Maximum number of idle threads	

<u>MaxThreads <i>number</i></u>	2048
Set the maximum number of worker threads	
<u>MaxThreadsPerChild <i>number</i></u>	64
Maximum number of threads per child process	
<u>MCacheMaxObjectCount <i>value</i></u>	1009
<u>MCacheMaxObjectSize <i>bytes</i></u>	10000
()	
<u>MCacheMaxStreamingBuffer <i>size in bytes</i></u>	100000
	MCacheMaxOb +
<u>MCacheMinObjectSize <i>bytes</i></u>	0
()	
<u>MCacheRemovalAlgorithm LRU GDSF</u>	GDSF
<u>MCacheSize <i>KBytes</i></u>	100
(KByte)	
<u>MetaDir <i>directory</i></u>	.web
CERN	
<u>MetaFiles on off</u>	off
CERN	
<u>MetaSuffix <i>suffix</i></u>	.meta
CERN	
<u>MimeMagicFile <i>file-path</i></u>	
Enable MIME-type determination based on file contents using the specified magic file	
<u>MinSpareServers <i>number</i></u>	5
Minimum number of idle child server processes	
<u>MinSpareThreads <i>number</i></u>	
Minimum number of idle threads available to handle request spikes	
<u>MMapFile <i>file-path [file-path] ...</i></u>	
<u>ModMimeUsePathInfo On Off</u>	Off
Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename	
<u>MultiviewsMatch</u>	NegotiatedOnly
<u>Any NegotiatedOnly Filters Handlers</u>	

[Handlers Filters]	
The types of files that will be included when searching for a matching file with MultiViews	
NameVirtualHost <i>addr[:port]</i>	
Designates an IP address for name-virtual hosting	
NoProxy <i>host [host] ...</i>	
Hosts, domains, or networks that will be connected to directly	
NumServers <i>number</i>	2
Total number of children alive at the same time	
NWSSLTrustedCerts <i>filename [filename] ...</i>	
List of additional client certificates	
NWSSLUpgradeable <i>[IP-address:]portnumber</i>	
Allows a connection to be upgraded to an SSL connection upon request	
Options <i>[+ -]option [[+ -]option] ...</i>	All
Configures what features are available in a particular directory	
Order <i>ordering</i>	Deny,Allow
Controls the default access state and the order in which Allow and Deny are evaluated.	
PassEnv <i>env-variable [env-variable] ...</i>	
PidFile <i>filename</i>	logs/httpd.pid
File where the server records the process ID of the daemon	
ProtocolEcho On Off	
echo	
<Proxy <i>wildcard-url</i>> ...</Proxy>	
Container for directives applied to proxied resources	
ProxyBadHeader <i>IsError Ignore StartBody</i>	IsError
Determines how to handle bad header lines in a response	
ProxyBlock <i>* word host domain [word host domain] ...</i>	
Words, hosts, or domains that are banned from being proxied	
ProxyDomain <i>Domain</i>	
Default domain name for proxied requests	
ProxyErrorOverride On Off	Off
Override error pages for proxied content	
ProxyFtpDirCharset <i>character set</i>	ISO-8859-1
Define the character set for proxied FTP listings	

ProxyIOBufferSize <i>bytes</i>	8192
Determine size of internal data throughput buffer	
<ProxyMatch <i>regex</i>> ...</ProxyMatch>	
Container for directives applied to regular-expression-matched proxied resources	
ProxyMaxForwards <i>number</i>	10
Maximum number of proxies that a request can be forwarded through	
ProxyPass [<i>path</i>] <i>! url</i>	
Maps remote servers into the local server URL-space	
ProxyPassReverse [<i>path</i>] <i>url</i>	
Adjusts the URL in HTTP response headers sent from a reverse proxied server	
ProxyPreserveHost On Off	Off
Use incoming Host HTTP request header for proxy request	
ProxyReceiveBufferSize <i>bytes</i>	0
Network buffer size for proxied HTTP and FTP connections	
ProxyRemote <i>match remote-server</i>	
Remote proxy used to handle certain requests	
ProxyRemoteMatch <i>regex remote-server</i>	
Remote proxy used to handle requests matched by regular expressions	
ProxyRequests On Off	Off
Enables forward (standard) proxy requests	
ProxyTimeout <i>seconds</i>	300
Network timeout for proxied requests	
ProxyVia On Off Full Block	Off
Information provided in the Via HTTP response header for proxied requests	
ReadmeName <i>filename</i>	
ReceiveBufferSize <i>bytes</i>	0
TCP receive buffer size	
Redirect [<i>status</i>] <i>URL-path URL</i>	
URL	
RedirectMatch [<i>status</i>] <i>regex URL</i>	
URL	
RedirectPermanent <i>URL-path URL</i>	
URL	
RedirectTemp <i>URL-path URL</i>	

URL	
RemoveCharset <i>extension</i> [<i>extension</i>] ...	
Removes any character set associations for a set of file extensions	
RemoveEncoding <i>extension</i> [<i>extension</i>] ...	
Removes any content encoding associations for a set of file extensions	
RemoveHandler <i>extension</i> [<i>extension</i>] ...	
Removes any handler associations for a set of file extensions	
RemoveInputFilter <i>extension</i> [<i>extension</i>] ...	
Removes any input filter associations for a set of file extensions	
RemoveLanguage <i>extension</i> [<i>extension</i>] ...	
Removes any language associations for a set of file extensions	
RemoveOutputFilter <i>extension</i> [<i>extension</i>] ...	
Removes any output filter associations for a set of file extensions	
RemoveType <i>extension</i> [<i>extension</i>] ...	
Removes any content type associations for a set of file extensions	
RequestHeader <i>set append add unset header</i> [<i>value</i>]	
HTTP	
Require <i>entity-name</i> [<i>entity-name</i>] ...	
Selects which authenticated users can access a resource	
RewriteBase <i>URL-path</i>	
Sets the base URL for per-directory rewrites	
RewriteCond <i>TestString CondPattern</i>	
Defines a condition under which rewriting will take place	
RewriteEngine <i>on off</i>	off
Enables or disables runtime rewriting engine	
RewriteLock <i>file-path</i>	
Sets the name of the lock file used for RewriteMap synchronization	
RewriteLog <i>file-path</i>	
Sets the name of the file used for logging rewrite engine processing	
RewriteLogLevel <i>Level</i>	0
Sets the verbosity of the log file used by the rewrite engine	
RewriteMap <i>MapName MapType:MapSource</i>	
Defines a mapping function for key-lookup	
RewriteOptions <i>Options</i>	MaxRedirects=10

Sets some special options for the rewrite engine	
RewriteRule <i>Pattern Substitution</i>	
Defines rules for the rewriting engine	
RLimitCPU <i>seconds max [seconds max]</i>	
Limits the CPU consumption of processes launched by Apache children	
RLimitMEM <i>bytes max [bytes max]</i>	
Limits the memory consumption of processes launched by Apache children	
RLimitNPROC <i>number max [number max]</i>	
Limits the number of processes that can be launched by processes launched by Apache ch	
Satisfy Any All	All
Interaction between host-level access control and user authentication	
ScoreBoardFile <i>file-path</i>	logs/apache_status
Location of the file used to store coordination data for the child processes	
Script <i>method cgi-script</i>	
CGI .	
ScriptAlias <i>URL-path file-path directory-path</i>	
URL CGI	
ScriptAliasMatch <i>regex file-path directory-path</i>	
URL CGI	
ScriptInterpreterSource Registry Registry-Strict Script	Script
Technique for locating the interpreter for CGI scripts	
ScriptLog <i>file-path</i>	
CGI	
ScriptLogBuffer <i>bytes</i>	1024
PUT POST	
ScriptLogLength <i>bytes</i>	10385760
CGI	
ScriptSock <i>file-path</i>	logs/cgisock
cgi	
SecureListen [<i>IP-address:</i>]portnumber <i>Certificate-Name [MUTUAL]</i>	
Enables SSL encryption for the specified port	
SendBufferSize <i>bytes</i>	0
TCP buffer size	

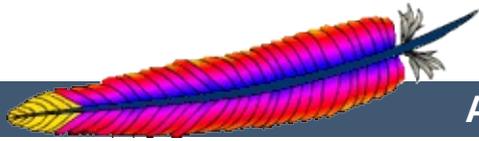
<u>ServerAdmin</u> <i>email-address</i>	
Email address that the server includes in error messages sent to the client	
<u>ServerAlias</u> <i>hostname [hostname] ...</i>	
Alternate names for a host used when matching requests to name-virtual hosts	
<u>ServerLimit</u> <i>number</i>	
Upper limit on configurable number of processes	
<u>ServerName</u> <i>fully-qualified-domain-name[:port]</i>	
Hostname and port that the server uses to identify itself	
<u>ServerPath</u> <i>URL-path</i>	
Legacy URL pathname for a name-based virtual host that is accessed by an incompatible b	
<u>ServerRoot</u> <i>directory-path</i>	<code>/usr/local/apache</code>
Base directory for the server installation	
<u>ServerSignature</u> <i>On Off EMail</i>	Off
Configures the footer on server-generated documents	
<u>ServerTokens</u>	Full
<u>Major Minor Min[imal] Prod[uctOnly] OS Full</u>	
Configures the Server HTTP response header	
<u>SetEnv</u> <i>env-variable value</i>	
<u>SetEnvIf</u> <i>attribute regex [!]<u>env-variable[=value]</u> [[!]<u>env-variable[=value]</u>] ...</i>	
<u>SetEnvIfNoCase</u> <i>attribute regex [!]<u>env-variable[=value]</u> [[!]<u>env-variable[=value]</u>] ...</i>	
<u>SetHandler</u> <i>handler-name None</i>	
Forces all matching files to be processed by a handler	
<u>SetInputFilter</u> <i>filter[:filter...]</i>	
Sets the filters that will process client requests and POST input	
<u>SetOutputFilter</u> <i>filter[:filter...]</i>	
Sets the filters that will process responses from the server	
<u>SSIEndTag</u> <i>tag</i>	<code>"-->"</code>
String that ends an include element	
<u>SSIErrorMsg</u> <i>message</i>	<code>"[an error occurred +</code>

Error message displayed when there is an SSI error	
<u>SSIStartTag tag</u>	"<!--#"
String that starts an include element	
<u>SSITimeFormat formatstring</u>	"%A, %d-%b-%Y %H:%M +"
Configures the format in which date strings are displayed	
<u>SSIUndefinedEcho string</u>	"(none)"
String displayed when an unset variable is echoed	
<u>SSLCACertificateFile file-path</u>	
File of concatenated PEM-encoded CA Certificates for Client Auth	
<u>SSLCACertificatePath directory-path</u>	
Directory of PEM-encoded CA Certificates for Client Auth	
<u>SSLCARevocationFile file-path</u>	
File of concatenated PEM-encoded CA CRLs for Client Auth	
<u>SSLCARevocationPath directory-path</u>	
Directory of PEM-encoded CA CRLs for Client Auth	
<u>SSLCertificateChainFile file-path</u>	
File of PEM-encoded Server CA Certificates	
<u>SSLCertificateFile file-path</u>	
Server PEM-encoded X.509 Certificate file	
<u>SSLCertificateKeyFile file-path</u>	
Server PEM-encoded Private Key file	
<u>SSLCipherSuite cipher-spec</u>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL handshake	
<u>SSLEngine on off</u>	off
SSL Engine Operation Switch	
<u>SSLHonorCipherOrder flag</u>	
Option to prefer the server's cipher preference order	
<u>SSLInsecureRenegotiation flag</u>	off
Option to enable support for insecure renegotiation	
<u>SSLMutex type</u>	none
Semaphore for internal mutual exclusion of operations	
<u>SSLOptions [+ -]option ...</u>	
Configure various SSL engine run-time options	

<u>SSLPassPhraseDialog <i>type</i></u>	builtin
Type of pass phrase dialog for encrypted private keys	
<u>SSLProtocol [+]-protocol ...</u>	all
Configure usable SSL protocol flavors	
<u>SSLProxyCACertificateFile <i>file-path</i></u>	
File of concatenated PEM-encoded CA Certificates for Remote Server Auth	
<u>SSLProxyCACertificatePath <i>directory-path</i></u>	
Directory of PEM-encoded CA Certificates for Remote Server Auth	
<u>SSLProxyCARevocationFile <i>file-path</i></u>	
File of concatenated PEM-encoded CA CRLs for Remote Server Auth	
<u>SSLProxyCARevocationPath <i>directory-path</i></u>	
Directory of PEM-encoded CA CRLs for Remote Server Auth	
<u>SSLProxyCipherSuite <i>cipher-spec</i></u>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL proxy handshake	
<u>SSLProxyEngine on off</u>	off
SSL Proxy Engine Operation Switch	
<u>SSLProxyMachineCertificateFile <i>filename</i></u>	
File of concatenated PEM-encoded client certificates and keys to be used by the proxy	
<u>SSLProxyMachineCertificatePath <i>directory</i></u>	
Directory of PEM-encoded client certificates and keys to be used by the proxy	
<u>SSLProxyProtocol [+]-protocol ...</u>	all
Configure usable SSL protocol flavors for proxy usage	
<u>SSLProxyVerify <i>level</i></u>	none
Type of remote server Certificate verification	
<u>SSLProxyVerifyDepth <i>number</i></u>	1
Maximum depth of CA Certificates in Remote Server Certificate verification	
<u>SSLRandomSeed <i>context source [bytes]</i></u>	
Pseudo Random Number Generator (PRNG) seeding source	
<u>SSLRequire <i>expression</i></u>	
Allow access only when an arbitrarily complex boolean expression is true	
<u>SSLRequireSSL</u>	
Deny access when SSL is not used for the HTTP request	
<u>SSLSessionCache <i>type</i></u>	none
Type of the global/inter-process SSL Session Cache	

<u>SSLSessionCacheTimeout</u> <i>seconds</i>	300
Number of seconds before an SSL session expires in the Session Cache	
<u>SSLUserName</u> <i>varname</i>	
Variable name to determine user name	
<u>SSLVerifyClient</u> <i>level</i>	none
Type of Client Certificate verification	
<u>SSLVerifyDepth</u> <i>number</i>	1
Maximum depth of CA Certificates in Client Certificate verification	
<u>StartServers</u> <i>number</i>	
Number of child server processes created at startup	
<u>StartThreads</u> <i>number</i>	
Number of threads created on startup	
<u>SuexecUserGroup</u> <i>User Group</i>	
CGI	
<u>ThreadLimit</u> <i>number</i>	
Sets the upper limit on the configurable number of threads per child process	
<u>ThreadsPerChild</u> <i>number</i>	
Number of threads created by each child process	
<u>ThreadStackSize</u> <i>number</i>	65536
Determine the stack size for each thread	
<u>TimeOut</u> <i>seconds</i>	300
Amount of time the server will wait for certain events before failing a request	
<u>TraceEnable</u> [<i>on off extended</i>]	on
Determines the behaviour on TRACE requests	
<u>TransferLog</u> <i>file pipe</i>	
<u>TypesConfig</u> <i>file-path</i>	conf/mime.types
The location of the mime.types file	
<u>UnsetEnv</u> <i>env-variable [env-variable] ...</i>	
<u>UseCanonicalName</u> <i>On Off DNS</i>	On
Configures how the server determines its own name and port	
<u>User</u> <i>unix-userid</i>	#-1
The userid under which the server will answer requests	
<u>UserDir</u> <i>directory-filename</i>	public_html

<u>VirtualDocumentRoot <i>interpolated-directory</i> none</u>	none
Dynamically configure the location of the document root for a given virtual host	
<u>VirtualDocumentRootIP <i>interpolated-directory</i> none</u>	none
Dynamically configure the location of the document root for a given virtual host	
<u><VirtualHost <i>addr[:port] [addr[:port]] ...> ...</i></u> <u></VirtualHost></u>	
Contains directives that apply only to a specific hostname or IP address	
<u>VirtualScriptAlias <i>interpolated-directory</i> none</u>	none
Dynamically configure the location of the CGI directory for a given virtual host	
<u>VirtualScriptAliasIP <i>interpolated-directory</i> none</u>	none
Dynamically configure the location of the cgi directory for a given virtual host	
<u>Win32DisableAcceptEx</u>	
Use accept() rather than AcceptEx() to accept network connections	
<u>XBitHack on off full</u>	off
Parse SSI directives in files with the execute bit set	



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



.



.

(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 .

leader

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

perchild

Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids

prefork

Implements a non-threaded, pre-forking web server

threadpool

Yet another experimental variant of the standard worker MPM

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_access](#)

Provides access control based on client hostname, IP address, or other characteristics of the client request.

[mod_actions](#)

CGI .

[mod_alias](#)

, URL

[mod_asis](#)

HTTP

[mod_auth](#)

User authentication using text files

[mod_auth_anon](#)

Allows "anonymous" user access to authenticated areas

[mod_auth_dbm](#)

Provides for user authentication using DBM files

[mod_auth_digest](#)

MD5 Digest Authentication .

[mod_auth_ldap](#)

Allows an LDAP directory to be used to store the database for HTTP Basic authentication.

[mod_autoindex](#)

Is Win32 dir

[mod_cache](#)

Content cache keyed to URIs.

[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_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_headers](#)

HTTP

[mod_imap](#)

(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_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_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_suexec](#)

CGI

[mod_unique_id](#)

[mod_userdir](#)

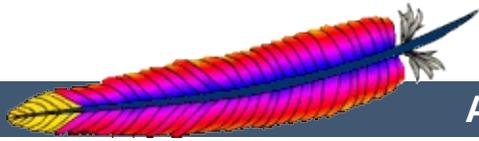
[mod_usertrack](#)

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

(FAQ)

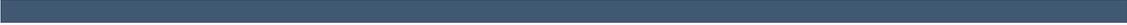
FAQ < <http://httpd.apache.org/docs/2.0/faq/> . ,

FAQ .

2.0 (FAQ) .

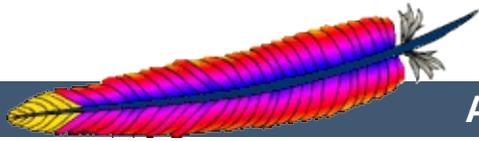
[1.3 FAQ](#) .





?

?



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



[Apache HTTP Server Version 2.0](#) .

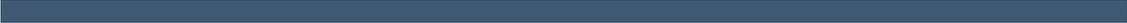


-
- [1.3 2.0](#)
 - [2.0](#)
 -



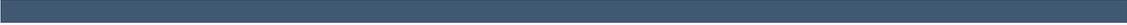
-
- [\[redacted\]](#)
 - [\[redacted\]](#)
 - [\[redacted\]](#)
 - [\[redacted\]](#)
 - [Directory, Location, Files](#)
 - [\[redacted\]](#)
 - [\[redacted\]](#)
 - [URL](#)
 - [\[redacted\]](#)
 - [\(DSO\)](#)
 - [\(content negotiation\)](#)
 - [\[redacted\]](#)
 - [\[redacted\]](#)
 - [\(MPM\)](#)
 - [\[redacted\]](#)
 - [\[redacted\]](#)
 - [\[redacted\]](#)
 - [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](#)
- [\[redacted\]](#)
- [\[redacted\]](#)



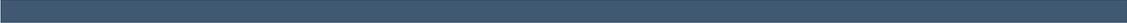
-
-

- [Microsoft Windows](#)
- [Microsoft Windows](#)
- [Novell NetWare](#)
- [HPUX](#)
- [EBCDIC](#)



-
- - [Manpage: httpd](#)
 - [Manpage: ab](#)
 - [Manpage: apachectl](#)
 - [Manpage: apxs](#)
 - [Manpage: configure](#)
 - [Manpage: dbmmanage](#)
 - [Manpage: htdigest](#)
 - [Manpage: htpasswd](#)
 - [Manpage: logresolve](#)
 - [Manpage: rotatelogs](#)
 - [Manpage: suexec](#)
 -





-
-
- FIN_WAIT_2
-
-
-



-
- [mod_auth](#)
 - [mod_auth_basic](#)
 - [mod_auth_digest](#)
 - [MPM](#)
 - [MPM beos](#)
 - [MPM leader](#)
 - [MPM netware](#)
 - [MPM os2](#)
 - [MPM perchild](#)
 - [MPM prefork](#)
 - [MPM threadpool](#)
 - [MPM winnt](#)
 - [MPM worker](#)
 - [mod_access](#)
 - [mod_actions](#)
 - [mod_alias](#)
 - [mod_asis](#)
 - [mod_auth](#)
 - [mod_auth_anon](#)
 - [mod_auth_dbm](#)
 - [mod_auth_digest](#)
 - [mod_auth_ldap](#)
 - [mod_autoindex](#)
 - [mod_cache](#)
 - [mod_cern_meta](#)
 - [mod_cgi](#)
 - [mod_cgid](#)
 - [mod_charset_lite](#)
 - [mod_dav](#)
 - [mod_dav_fs](#)
 - [mod_deflate](#)

- [mod_dir](#)
- [mod_disk_cache](#)
- [mod_dumpio](#)
- [mod_echo](#)
- [mod_env](#)
- [mod_example](#)
- [mod_expires](#)
- [mod_ext_filter](#)
- [mod_file_cache](#)
- [mod_headers](#)
- [mod_imap](#)
- [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_connect](#)
- [mod_proxy_ftp](#)
- [mod_proxy_http](#)
- [mod_rewrite](#)
- [mod_setenvif](#)
- [mod_so](#)
- [mod_speling](#)
- [mod_ssl](#)
- [mod_status](#)
- [mod_suexec](#)

- [mod_unique_id](#)
- [mod_userdir](#)
- [mod_usertrack](#)
- [mod_version](#)
- [mod_vhost_alias](#)



-

- [Apache API](#)

- [APR](#)

- [Apache 2.0](#)

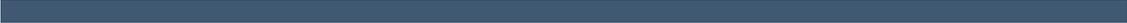
- [Apache 2.0 \(hook\)](#)

- [Apache 1.3 Apache 2.0](#)

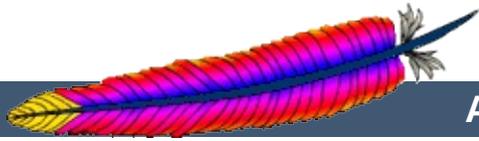
- [Apache 2.0](#)

- [Apache 2.0](#)





-
- 
- 
- 



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

--



.



[httpd](#)

[apachectl](#)

[ab](#)

[apxs](#)

(APache eXtenSion tool)

[configure](#)

[dbmmanage](#)

basic authentication DBM

[htdigest](#)

digest authentication

[htpasswd](#)

basic authentication

[logresolve](#)

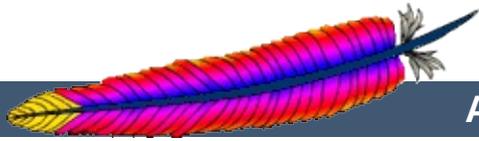
IP-

[rotatelogs](#)

[suexec](#)

(Switch User For Exec)

manpage .



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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.



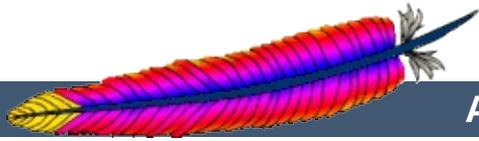
-
- [Introduction](#)
 - [Compatibility](#)
 - [How-To](#)
 - [Frequently Asked Questions](#)
 - [Glossary](#)



Extensive documentation on the directives and environment variables provided by this module is provided in the [mod_ssl reference documentation](#).

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



(Virtual Host) (, www.company1.co
 www.company2.com) . IP
 based)" IP " (name-based) " .

IP . 1.1 IP
 (host-based) IP (non-IP virtual hosts) .

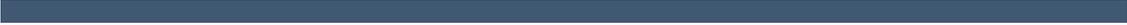
1.3

mod vhost alias

```

  IP
  
```





- (IP)
- IP (IP)
-
- (file descriptor) (,)
-
-

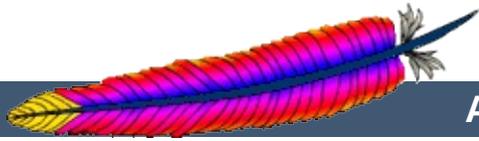


- [<VirtualHost>](#)
- [NameVirtualHost](#)
- [ServerName](#)
- [ServerAlias](#)
- [ServerPath](#)

-S ., :

```
/usr/local/apache2/bin/httpd -S
```

([httpd](#) .) . IP



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

Developer Documentation for Apache 2.0

Many of the documents on these Developer pages are lifted from Apache 1.3's documentation. While they are all being updated to Apache 2.0, 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 mailing list.



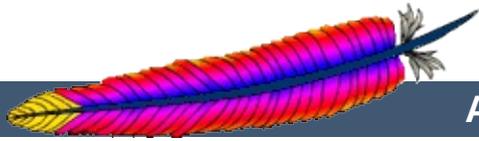
- [Apache 1.3 API Notes](#)
- [Apache 2.0 Hook Functions](#)
- [Request Processing in Apache 2.0](#)
- [How filters work in Apache 2.0](#)
- [Converting Modules from Apache 1.3 to Apache 2.0](#)
- [Debugging Memory Allocation in APR](#)
- [Documenting Apache 2.0](#)
- [Apache 2.0 Thread Safety Issues](#)



-
- Tools provided by Ian Holsman:
 - [Apache 2 cross reference](#)
 - [Autogenerated Apache 2 code documentation](#)
 - Module Development Tutorials by Kevin O'Donnell
 - [Integrating a module into the Apache build system](#)
 - [Handling configuration directives](#)
 - [Some notes on Apache module development by Ryan Bloom](#)

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

Apache Miscellaneous Documentation

Below is a list of additional documentation pages that apply to the Apache web server development project.

Warning

Some of the documents below have not been fully 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.

[How to use XSSI and Negotiation for custom ErrorDocuments](#)

Describes a solution which uses XSSI and negotiation to custom-tailor the Apache ErrorDocuments to taste, adding the advantage of returning internationalized versions of the error messages depending on the client's language preferences.

[File Descriptor use in Apache](#)

Describes how Apache uses file descriptors and talks about various limits imposed on the number of descriptors available by various operating systems.

[FIN_WAIT_2](#)

A description of the causes of Apache processes going into the FIN_WAIT_2 state, and what you can do about it.

[Known Client Problems](#)

A list of problems in HTTP clients which can be mitigated by Apache.

[Performance Notes - Apache Tuning](#)

Notes about how to (run-time and compile-time) configure Apache for highest performance. Notes explaining why Apache does some things, and why it doesn't do other things (which make it slower/faster).

[Security Tips](#)

Some "do"s - and "don't"s - for keeping your Apache web site secure.

[URL Rewriting Guide](#)

This document supplements the [mod_rewrite reference documentation](#). It describes how one can use Apache's [mod_rewrite](#) to solve typical URL-based problems webmasters are usually confronted with in practice.

[Apache Tutorials](#)

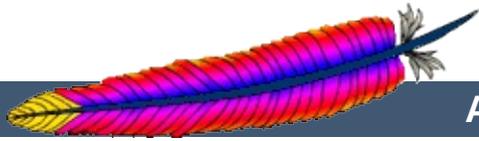
A list of external resources which help to accomplish common tasks with the Apache HTTP server.

[Relevant Standards](#)

This document acts as a reference page for most of the relevant standards that Apache follows.

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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



| | [FAQ](#) | |



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)



Microsoft Windows 2.0 , , .

: [Microsoft Windows](#)

.

: [Microsoft Windows](#)



Novell NetWare

Novell NetWare 5.1 2.0

: [Novell NetWare](#)

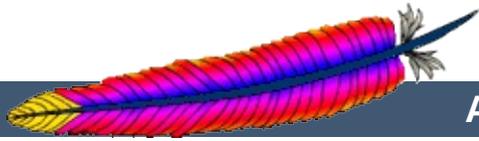
EBCDIC

1.3 EBCDIC

(-ASCII)

: 2.0 . ,
.

: [EBCDIC](#)



| | [FAQ](#) | |

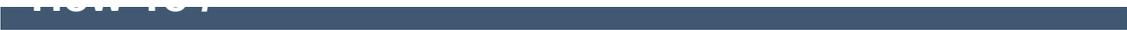


Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

How-To /





(authentication) . (authorization)

.
:
: [.htaccess](#)

CGI

CGI (Common Gateway Interface) CGI CGI
, () .
. CGI , CGI .
:
: [CGI:](#)

.htaccess

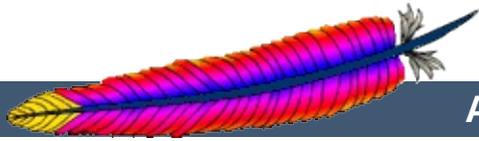
.htaccess .
, .
:
: [.htaccess](#)

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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

suexec -

```
CGI                                suexec .  
.      root                        suexec setuid  
root      .
```

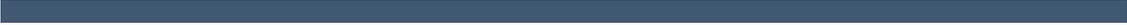
suexec suexec

(<http://httpd.apache.org/docs/2.0/suexec.html>) .



suexec -V

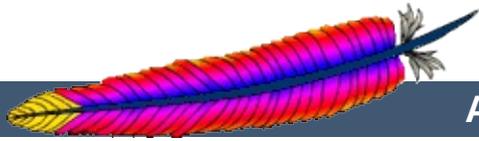




-v

root suexec .

.



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

Apache mod_rewrite

``The great thing about mod_rewrite is it gives you all the configurability and flexibility of Sendmail. The downside to mod_rewrite is that it gives you all the configurability and flexibility of Sendmail.''

-- Brian Behlendorf
Apache Group

`` Despite the tons of examples and docs, mod_rewrite is voodoo. Damned cool voodoo, but still voodoo. ''

-- Brian Moore
bem@news.cmc.net

Welcome to mod_rewrite, the Swiss Army Knife of URL manipulation!

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, for instance server variables, environment variables, HTTP headers, time stamps and even external database lookups in various formats can be used to achieve 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 even 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.

But all this functionality and flexibility has its drawback: complexity. So don't expect to understand this entire module in just one day.



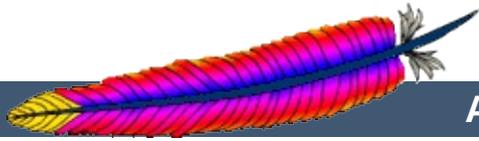
- [Introduction](#)
- [Technical details](#)
- [Practical solutions to common problems](#)
- [Glossary](#)



Extensive documentation on the directives provided by this module is provided in the [mod_rewrite reference documentation](#).

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

URL Rewriting Guide

This document supplements the [mod_rewrite reference documentation](#). It describes how one can use Apache's [mod_rewrite](#) to solve typical URL-based problems with which webmasters are commonly confronted. We give detailed descriptions on how to solve each problem by configuring URL rewriting rulesets.

ATTENTION: Depending on your server configuration it may be necessary to slightly change the examples for your situation, e.g. adding the [PT] flag when additionally using [mod_alias](#) and [mod_userdir](#), etc. Or rewriting a ruleset to fit in `.htaccess` context instead of per-server context. Always try to understand what a particular ruleset really does before you use it. This avoids many problems.

See also

[Module documentation](#)
[mod_rewrite introduction](#)
[Technical details](#)



Description:

On some web servers there are more than one URL for a resource. Usually there are canonical URLs (which should be actually used and distributed) and those which are just shortcuts, internal ones, etc. Independent of which URL the user supplied with the request he should finally see the canonical one only.

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 `/~user` by the canonical `/u/user` and fix a missing trailing slash for `/u/user`.

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



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:

For sites running on a port other than 80:

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

And for a site running on port 80

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



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/e/www/
```



Description:

The vast majority of "trailing slash" problems can be dealt with using the techniques discussed in the [FAQ entry](#). However, occasionally, there is a need to use `mod_rewrite` to handle a case where a missing trailing slash causes a URL to fail. This can happen, for example, after a series of complex rewrite rules.

Solution:

The solution to this subtle problem is to let the server add the trailing slash automatically. To do this correctly we have to use an external redirect, so the browser correctly requests subsequent images etc. If we only did a internal rewrite, this would only work for the directory page, but would go wrong when any images are included into this page with relative URLs, because the browser would request an in-lined object. For instance, a request for `image.gif` in `/~quux/foo/index.html` would become `/~quux/image.gif` without the external redirect!

So, to do this trick we write:

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

Alternately, you can put the following in a top-level `.htaccess` file in the content directory. But note that this creates some processing overhead.

```
RewriteEngine on
RewriteBase /~quux/
RewriteCond %{REQUEST_FILENAME} -d
```

```
RewriteRule ^(.+[^/])$ $1/ [R]
```



Description:

Many webmasters have asked for a solution to the following situation: They wanted to redirect just all homedirs on a webserver to another webserver. They usually need such things when establishing a newer webserver which will replace the old one over time.

Solution:

The solution is trivial with `mod_rewrite`. On the old webserver we just redirect all `/~user/anypath` URLs to `http://newserver/~user/anypath`.

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



Description:

Sometimes it is necessary to let the webserver search for pages in more than one directory. Here MultiViews or other techniques cannot help.

Solution:

We program a explicit ruleset which searches for the files in the directories.

```
RewriteEngine on

# first try to find it in custom/...
# ...and if found stop and be happy:
RewriteCond          /your/docroot/dir1/{REQUEST_FILENAME}
RewriteRule  ^(.+) /your/docroot/dir1/$1  [L]

# second try to find it in pub/...
# ...and if found stop and be happy:
RewriteCond          /your/docroot/dir2/{REQUEST_FILENAME}
RewriteRule  ^(.+) /your/docroot/dir2/$1  [L]

# else go on for other Alias or ScriptAlias directives,
# etc.
RewriteRule  ^(.+) - [PT]
```



Description:

Perhaps you want to keep status information between requests and use the URL to encode it. But you don't want to use a CGI wrapper for all pages just to strip out this information.

Solution:

We use a rewrite rule to strip out the status information and remember it via an environment variable which can be later dereferenced from within XSSI or CGI. This way a URL `/foo/S=java/bar/` gets translated to `/foo/bar/` and the environment variable named `STATUS` is set to the value `"java"`.

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



Description:

Assume that you want to provide `www.username.host.domain.com` for the homepage of `username` via just DNS A records to the same machine and without any virtualhosts on this machine.

Solution:

For HTTP/1.0 requests there is no solution, but for HTTP/1.1 requests which contain a Host: HTTP header we can use the following ruleset to rewrite `http://www.username.host.com/anypath` internally to `/home/username/anypath`:

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



Description:

We want to redirect homedir URLs to another webserver `www.somewhere.com` when the requesting user does not stay in the local domain `ourdomain.com`. This is sometimes used in virtual host contexts.

Solution:

Just a rewrite condition:

```
RewriteEngine on
RewriteCond  %{REMOTE_HOST}  !^.+\.ourdomain\.com$
RewriteRule  ^(/~.+ )        http://www.somewhere.com/$1 [R
```



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.



Description:

When tricks like time-dependent content should happen a lot of webmasters still use CGI scripts which do for instance redirects to specialized pages. How can it be done via [mod_rewrite](#)?

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
RewriteRule    ^foo\.html$              foo.night.html
```

This provides the content of `foo.day.html` under the URL `foo.html` from `07:00-19:00` and at the remaining time the contents of `foo.night.html`. Just a nice feature for a homepage...



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 .phtml?

Solution:

We just 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.phtml
# when and only when document.phtml exists
# but no longer document.html
RewriteEngine on
RewriteBase /~quux/
# parse out basename, but remember the fact
RewriteRule ^(.*)\.html$ $1 [C,E=wasHTML]
# rewrite to document.phtml if exists
RewriteCond %{REQUEST_FILENAME}.phtml -f
RewriteRule ^(.*)$ $1.phtml [S=1]
# else reverse the previous basename cutout
RewriteCond %{ENV:wasHTML} ^yes$
RewriteRule ^(.*)$ $1.html
```



From Old to New (intern)

Description:

Assume we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. Actually we want that users of the old URL even not recognize that the pages was renamed.

Solution:

We rewrite the old URL to the new one internally via the following rule:

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

From Old to New (extern)

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

From Static to Dynamic

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 correct MIME-type so it gets really run as a CGI-script. 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 [T=application/x-httpd-
```



Blocking of Robots

Description:

How can we block a really annoying robot from retrieving pages of a specific webarea? A `/robots.txt` file containing entries of the "Robot Exclusion Protocol" is typically not enough to get rid of such a robot.

Solution:

We use a ruleset which forbids the URLs of the webarea `/~quux/foo/arc/` (perhaps a very deep directory indexed area where the robot traversal would create big server load). We have to make sure that we forbid access only to the particular robot, i.e. just forbidding the host where the robot runs is not enough. This would block users from this host, too. We accomplish this by also matching the User-Agent HTTP header information.

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

Blocked Inline-Images

Description:

Assume we have under `http://www.quux-corp.de/~quux/` some pages with inlined GIF graphics. These graphics are nice, so others directly incorporate them via hyperlinks to their pages. We don't like this practice because it adds useless traffic to our server.

Solution:

While we cannot 100% protect the images from inclusion, we

can at least restrict the cases where the browser sends a HTTP Referer header.

```
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$ -
```

Proxy Deny

Description:

How can we forbid a certain host or even a user of a special host from using the Apache proxy?

Solution:

We first have to make sure `mod_rewrite` is below(!) `mod_proxy` in the Configuration file when compiling the Apache webserver. This way it gets called *before* `mod_proxy`. Then we configure the following for a host-dependent deny...

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

...and this one for a user@host-dependent deny:

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



External Rewriting Engine

Description:

A FAQ: How can we solve the FOO/BAR/QUUX/etc. problem?
There seems no solution by the use of `mod_rewrite`...

Solution:

Use an external `RewriteMap`, i.e. a program which acts like a `RewriteMap`. It is run once on startup of Apache receives the requested URLs on STDIN and has to put the resulting (usually rewritten) URL on STDOUT (same order!).

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

```
#!/path/to/perl

#  disable buffered I/O which would lead
#  to deadloops for the Apache server
$| = 1;

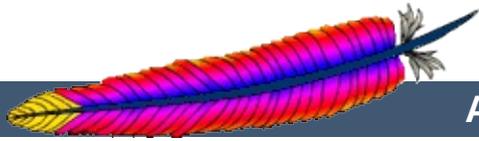
#  read URLs one per line from stdin and
#  generate substitution URL on stdout
while (<>) {
    s|^foo/|bar/|;
    print $_;
}
```

This is a demonstration-only example and just rewrites all URLs `/~quux/foo/...` to `/~quux/bar/...`. Actually you can program whatever you like. But notice that while such

maps can be **used** also by an average user, only the system administrator can **define** it.

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

URL Rewriting Guide - Advanced topics

This document supplements the [mod_rewrite reference documentation](#). It describes how one can use Apache's [mod_rewrite](#) to solve typical URL-based problems with which webmasters are commonly confronted. We give detailed descriptions on how to solve each problem by configuring URL rewriting rulesets.

ATTENTION: Depending on your server configuration it may be necessary to adjust the examples for your situation, e.g., adding the [PT] flag if using [mod_alias](#) and [mod_userdir](#), etc. Or rewriting a ruleset to work in .htaccess context instead of per-server context. Always try to understand what a particular ruleset really does before you use it; this avoids many problems.

See also

[Module documentation](#)
[mod_rewrite introduction](#)
[Technical details](#)



Description:

We want to create a homogeneous and consistent URL layout across all WWW servers on an Intranet web cluster, i.e., all URLs (by definition server-local and thus server-dependent!) become server *independent*! What we want is to give the WWW namespace a single consistent layout: no URL should refer to any particular target server. The cluster itself should connect users automatically to a physical target host as needed, invisibly.

Solution:

First, the knowledge of the target servers comes from (distributed) external maps which contain information on where our users, groups, and entities reside. They have the form:

```
user1  server_of_user1
user2  server_of_user2
:      :
```

We put them into files `map.xxx-to-host`. Second we need to instruct all servers to redirect URLs of the forms:

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

to

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

when any URL path need not be valid on every server. 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      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\
```



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, `/~foo/anypath` is `/home/f/foo/.www/anypath` while `/~bar/anypath` is `/home/b/bar/.www/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/.www$3
```



Description:

This really is a hardcore example: a killer application which heavily uses per-directory RewriteRules to get a smooth look and feel on the Web while its data structure is never touched or adjusted. Background: *net.sw* is my archive of freely available Unix software packages, which I started to collect in 1992. It is both my hobby and job to do this, because while I'm studying computer science I have also worked for many years as a system and network administrator in my spare time. Every week I need some sort of software so I created a deep hierarchy of directories where I stored the packages:

```
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/
```

In July 1996 I decided to make this archive public to the world via a nice Web interface. "Nice" means that I wanted to offer an interface where you can browse directly through the

archive hierarchy. And "nice" means that I didn't want to change anything inside this hierarchy - not even by putting some CGI scripts at the top of it. Why? Because the above structure should later be accessible via FTP as well, and I didn't want any Web or CGI stuff mixed in there.

Solution:

The solution has two parts: The first is a set of CGI scripts which create all the pages at all directory levels on-the-fly. I put them under `/e/netsw/.www/` as follows:

```
-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
```

The `DATA/` subdirectory holds the above directory structure, *i.e.* the real **net.sw** stuff, and gets automatically updated via `rdist` from time to time. The second part of the problem remains: how to link these two structures together into one smooth-looking URL tree? We want to hide the `DATA/` directory from the user while running the appropriate CGI scripts for the various URLs. Here is the solution: first I put the following into the per-directory configuration file in the [DocumentRoot](#) of the server to rewrite the public URL path

[/net.sw/](#) to the internal path `/e/netsw`:

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

The first rule is for requests which miss the trailing slash! The second rule does the real thing. And then comes the killer configuration which stays in the per-directory config file `/e/netsw/.www/.wwwacl`:

```
Options ExecCGI FollowSymLinks Includes MultiViews

RewriteEngine on

# we are reached via /net.sw/ prefix
RewriteBase /net.sw/

# first we rewrite the root dir to
# the handling cgi script
RewriteRule ^$ netsw-home.cgi [L]
RewriteRule ^index\.html$ netsw-home.cgi [L]

# strip out the subdirs when
# the browser requests us from perdir pages
RewriteRule ^.+/(netsw-[^\.]+/)+$ $1 [L]

# and now break the rewriting for local files
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]
```

```
# anything else is a subdir which gets handled
# by another cgi script
RewriteRule    !^netsw-lsdir\.cgi.*      -                [C
RewriteRule    (.*)                      netsw-lsdir.cgi/$1
```

Some hints for interpretation:

1. Notice the L (last) flag and no substitution field ('-') in the fourth part
2. Notice the ! (not) character and the C (chain) flag at the first rule in the last part
3. Notice the catch-all pattern in the last rule



Description:

A typical FAQ about URL rewriting is how to redirect failing requests on webserver A to webserver B. Usually this is done via [ErrorDocument](#) CGI scripts in Perl, but there is also a [mod_rewrite](#) solution. But note that this performs more poorly than using an [ErrorDocument](#) CGI script!

Solution:

The first solution has the best performance but less flexibility, and is less safe:

```
RewriteEngine on
RewriteCond    /your/docroot/%{REQUEST_FILENAME} !-f
RewriteRule    ^(.+)                                http://webse
```

The problem here is that this will only work for pages inside the [DocumentRoot](#). While you can add more Conditions (for instance to also handle homedirs, etc.) there is a better variant:

```
RewriteEngine on
RewriteCond    %{REQUEST_URI} !-U
RewriteRule    ^(.+)                http://webserverB.dom/$1
```

This uses the URL look-ahead feature of [mod_rewrite](#). The result is that this will work for all types of URLs and is safe. But it does have a performance impact on the web server, because for every request there is one more internal subrequest. So, if your web server runs on a powerful CPU, use this one. If it is a slow machine, use the first approach or better an [ErrorDocument](#) CGI script.



Description:

Do you know the great CPAN (Comprehensive Perl Archive Network) under <http://www.perl.com/CPAN>? CPAN automatically redirects browsers to one of many FTP servers around the world (generally one near the requesting client); each server carries a full CPAN mirror. This is effectively an FTP access multiplexing service. CPAN runs via CGI scripts, but how could a similar approach be implemented via [mod_rewrite](#)?

Solution:

First we notice that as of version 3.0.0, [mod_rewrite](#) can also use the "ftp:" scheme on redirects. And second, the location approximation can be done by a [RewriteMap](#) over the top-level domain of the client. With a tricky chained ruleset we can use this top-level domain as a key to our multiplexing map.

```
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##
```



Browser Dependent Content

Description:

At least for important top-level pages it is sometimes necessary to provide the optimum of browser dependent content, i.e., one has to provide one version for current browsers, a different version for the Lynx and text-mode browsers, and another for other browsers.

Solution:

We cannot use content negotiation because the browsers do not provide their type in that form. Instead we have to act on the HTTP header "User-Agent". The following config does the following: If the HTTP header "User-Agent" begins with "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]
```

Dynamic Mirror

Description:

Assume there are nice web pages on remote hosts we want

to bring into our namespace. For FTP servers we would use the `mirror` program which actually maintains an explicit up-to-date copy of the remote data on the local machine. For a web server we could use the program `webcopy` which runs via HTTP. But both techniques have a major drawback: The local copy is always only as up-to-date as the last time we ran the program. It would be much better if the mirror was not a static one we have to establish explicitly. Instead we want a dynamic mirror with data which gets updated automatically as needed on the remote host(s).

Solution:

To provide this feature we map the remote web page or even the complete remote web area to our namespace by the use of the *Proxy Throughput* feature (flag [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
```

Reverse Dynamic Mirror

Description:

...

Solution:

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

Retrieve Missing Data from Intranet

Description:

This is a tricky way of virtually running a corporate (external) Internet web server (`www.quux-corp.dom`), while actually keeping and maintaining its data on an (internal) Intranet web server (`www2.quux-corp.dom`) which is protected by a firewall. The trick is that the external web server retrieves the requested data on-the-fly from the internal one.

Solution:

First, we must make sure that our firewall still protects the internal web server and only the external web server is allowed to retrieve data from it. On a packet-filtering firewall, for instance, we could configure a firewall ruleset like the following:

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

Just adjust it to your actual configuration syntax. Now we can establish the [mod_rewrite](#) rules which request the missing data in the background through the proxy throughput feature:

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

Load Balancing

Description:

Suppose we want to load balance the traffic to `www.foo.com` over `www[0-5].foo.com` (a total of 6 servers). How can this

be done?

Solution:

There are many possible solutions for this problem. We will first discuss a common DNS-based method, and then one based on [mod_rewrite](#):

1. DNS Round-Robin

The simplest method for load-balancing is to use DNS round-robin. Here you just configure `www[0-9].foo.com` as usual in your DNS with A (address) records, e.g.,

```
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
```

Then you additionally add the following entries:

```
www     IN  A      1.2.3.1
www     IN  A      1.2.3.2
www     IN  A      1.2.3.3
www     IN  A      1.2.3.4
www     IN  A      1.2.3.5
```

Now when `www.foo.com` gets resolved, BIND gives out `www0-www5` - but in a permuted (rotated) order every time. This way the clients are spread over the various servers. But notice that this is not a perfect load balancing scheme, because DNS resolutions are cached by clients and other nameservers, so once a client has

resolved `www.foo.com` to a particular `wwwN.foo.com`, all its subsequent requests will continue to go to the same IP (and thus a single server), rather than being distributed across the other available servers. But the overall result is okay because the requests are collectively spread over the various web servers.

2. DNS Load-Balancing

A sophisticated DNS-based method for load-balancing is to use the program `lbname` which can be found at <http://www.stanford.edu/~schemers/docs/lbname/lbname>. It is a Perl 5 program which, in conjunction with auxiliary tools, provides real load-balancing via DNS.

3. Proxy Throughput Round-Robin

In this variant we use `mod_rewrite` and its proxy throughput feature. First we dedicate `www0.foo.com` to be actually `www.foo.com` by using a single

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

entry in the DNS. Then we convert `www0.foo.com` to a proxy-only server, i.e., we configure this machine so all arriving URLs are simply passed through its internal proxy to one of the 5 other servers (`www1-www5`). To accomplish this we first establish a ruleset which contacts a load balancing script `lb.pl` for all URLs.

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

Then we write `lb.pl`:

```

#!/path/to/perl
##
## lb.pl -- load balancing script
##

$| = 1;

$name     = "www";      # the hostname base
$first    = 1;          # the first server (not 0 here, bec
$last     = 5;          # the last server in the round-robi
$domain   = "foo.dom"; # the domainname

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

##EOF##

```

A last notice: Why is this useful? Seems like `www0.foo.com` still is overloaded? The answer is yes, it is overloaded, but with plain proxy throughput requests, only! All SSI, CGI, ePerl, etc. processing is handled done on the other machines. For a complicated site, this may work well. The biggest risk here is that `www0` is now a single point of failure -- if it crashes, the other servers are inaccessible.

4. Dedicated Load Balancers

There are more sophisticated solutions, as well. Cisco, F5, and several other companies sell hardware load

balancers (typically used in pairs for redundancy), which offer sophisticated load balancing and auto-failover features. There are software packages which offer similar features on commodity hardware, as well. If you have enough money or need, check these out. The [lb-l mailing list](#) is a good place to research.

New MIME-type, New Service

Description:

On the net there are many nifty CGI programs. But their usage is usually boring, so a lot of webmasters don't use them. Even Apache's Action handler feature for MIME-types is only appropriate when the CGI programs don't need special URLs (actually PATH_INFO and QUERY_STRING) as their input. First, let us configure a new file type with extension `.scgi` (for secure CGI) which will be processed by the popular `cgwrap` program. The problem here is that for instance if we use a Homogeneous URL Layout (see above) a file inside the user homedirs might have a URL like `/u/user/foo/bar.scgi`, but `cgwrap` needs URLs in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uqe]/([^\s/]+)/\.\www/(.+)\.scgi(.*). . .  
. . . /internal/cgi/user/cgwrap/~$1/$2.scgi$3 [NS,T=applicat
```

Or assume we have some more nifty programs: `wwwlog` (which displays the `access.log` for a URL subtree) and `wwidx` (which runs Glimpse on a URL subtree). We have to provide the URL area to these programs so they know which area they are really working with. But usually this is complicated, because they may still be requested by the alternate URL form, i.e., typically we would run the `swidx`

program from within `/u/user/foo/` via hyperlink to

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

which is ugly, because we have to hard-code **both** the location of the area **and** the location of the CGI inside the hyperlink. When we have to reorganize, we spend a lot of time changing the various hyperlinks.

Solution:

The solution here is to provide a special new URL format which automatically leads to the proper CGI invocation. We configure the following:

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

Now the hyperlink to search at `/u/user/foo/` reads only

```
HREF="*"
```

which internally gets automatically transformed to

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

The same approach leads to an invocation for the access log CGI program when the hyperlink `:log` gets used.

On-the-fly Content-Regeneration

Description:

Here comes a really esoteric feature: Dynamically generated but statically served pages, i.e., pages should be delivered as

pure static pages (read from the filesystem and just passed through), but they have to be generated dynamically by the web server if missing. This way you can have CGI-generated pages which are statically served unless an admin (or a cron job) removes the static contents. Then the contents gets refreshed.

Solution:

This is done via the following ruleset:

```
RewriteCond %{REQUEST_FILENAME}    !-s
RewriteRule ^page\.html$           page.cgi    [T=application/
```

Here a request for `page.html` leads to an internal run of a corresponding `page.cgi` if `page.html` is missing or has filesize null. The trick here is that `page.cgi` is a CGI script which (additionally to its `STDOUT`) writes its output to the file `page.html`. Once it has completed, the server sends out `page.html`. When the webmaster wants to force a refresh of the contents, he just removes `page.html` (typically from cron).

Document With Autorefresh

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

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 "&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##

```

Mass Virtual Hosting

Description:

The [VirtualHost](#) feature of Apache is nice and works great when you just have a few dozen virtual hosts. But when you are an ISP and have hundreds of virtual hosts, this feature is suboptimal.

Solution:

To provide this feature we map the remote web page or even the complete remote web area to our namespace using the *Proxy Throughput* feature (flag [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
##
:
# use the canonical hostname on redirects, etc.
UseCanonicalName on

:
# add the virtual host in front of the CLF-format
CustomLog /path/to/access_log "%{VHOST}e %h %l %u %t \"%r\"
:

# enable the rewriting engine in the main server
RewriteEngine on

# define two maps: one for fixing the URL and one which de
# the available virtual hosts with their corresponding
# DocumentRoot.
RewriteMap lowercase int:tolower
RewriteMap vhost txt:/path/to/vhost.map

# Now do the actual virtual host mapping
# via a huge and complicated single rule:
#
# 1. make sure we don't map for common locations
RewriteCond %{REQUEST_URI} !^/commonurl1/. *
RewriteCond %{REQUEST_URI} !^/commonurl2/. *
:
RewriteCond %{REQUEST_URI} !^/commonurlN/. *
#
```

```
# 2. make sure we have a Host header, because
#     currently our approach only supports
#     virtual hosting through this header
RewriteCond  %{HTTP_HOST}  !^$
#
# 3. lowercase the hostname
RewriteCond  ${lowercase:%{HTTP_HOST}|NONE}  ^(.+)$
#
# 4. lookup this hostname in vhost.map and
#     remember it only when it is a path
#     (and not "NONE" from above)
RewriteCond  ${vhost:%1}  ^(/.*)$
#
# 5. finally we can map the URL to its docroot location
#     and remember the virtual host for logging purposes
RewriteRule  ^/(.*)$  %1/$1  [E=VHOST:${lowercase:%{HTTP_H
:

```



Host Deny

Description:

How can we forbid a list of externally configured hosts from using our server?

Solution:

For Apache >= 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]
```

For Apache <= 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
##
##  ATTENTION! This is a map, not a list, even when we treat
##              mod_rewrite parses it for key/value pairs, so
##              dummy value "-" must be present for each entr
##
```

```
193.102.180.41 -  
bsdti1.sdm.de -  
192.76.162.40 -
```

Proxy Deny

Description:

How can we forbid a certain host or even a user of a special host from using the Apache proxy?

Solution:

We first have to make sure `mod_rewrite` is below(!) `mod_proxy` in the Configuration file when compiling the Apache web server. This way it gets called *before* `mod_proxy`. Then we configure the following for a host-dependent deny...

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

...and this one for a user@host-dependent deny:

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

Special Authentication Variant

Description:

Sometimes very special authentication is needed, for instance authentication which checks for a set of explicitly configured users. Only these should receive access and without explicit prompting (which would occur when using Basic Auth via `mod_auth`).

Solution:

We use a list of rewrite conditions to exclude all except our friends:

```
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-based Deflector

Description:

How can we program a flexible URL Deflector which acts on the "Referer" HTTP header and can be configured with as many referring pages as we like?

Solution:

Use the following really tricky ruleset...

```
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]
```

... in conjunction with a corresponding rewrite map:

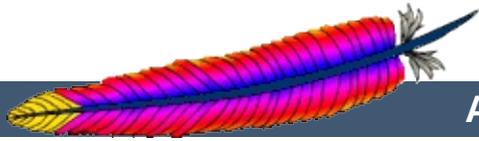
```
##
## deflector.map
##
```

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

This automatically redirects the request back to the referring page (when "-" is used as the value in the map) or to a specific URL (when an URL is specified in the map as the second argument).

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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](#)

[Practical solutions to common problems](#)



Internal Processing

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.

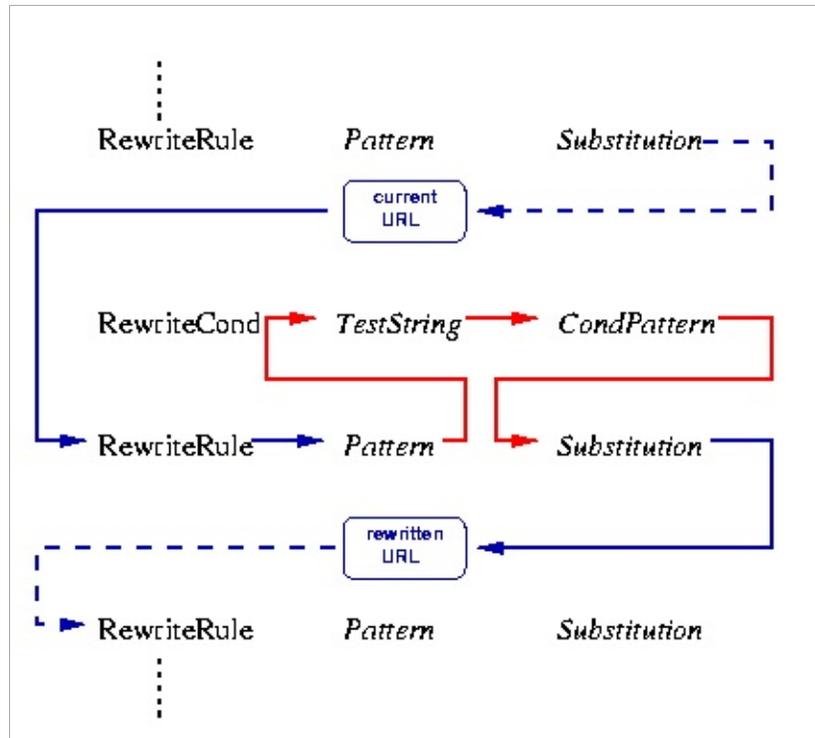
Don't forget these two points!



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.



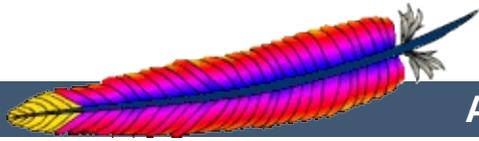
Figure

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*.

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

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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [FAQ](#)

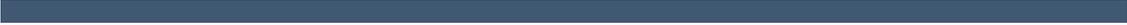
(FAQ)

FAQ < <http://httpd.apache.org/docs/2.0/faq/> .

2.0 (FAQ) .

[1.3 FAQ](#) .





?

?



- [" ... ? ... ?"](#)
- [?](#)

" ... ? ... ?"

:

(errorlog) !

(
/usr/local/apache2/logs/error_log,
[ErrorLog](#) .

[FAQ](#) !

FAQ

(The Apache Group)
(open) (closed)

"

USENET :

- [comp.infosystems.www.servers.unix](#) [[news](#)] [[google](#)]
- [comp.infosystems.www.servers.ms-windows](#) [[news](#)]
[[google](#)]
- [comp.infosystems.www.authoring.cgi](#) [[news](#)] [[google](#)]

- [Invalid argument: core_output_filter: writing data to the network](#)
- [AcceptEx failed](#)
- [Premature end of script headers](#)

Invalid argument: core_output_filter: writing data to the network

```

sendfile ,
sendfile
(error log) 0 0
sendfile
sendfile EnableSendfile .,
EnableMMAP .

```

AcceptEx Failed

```

win32 AcceptEx , Win32DisableAcc

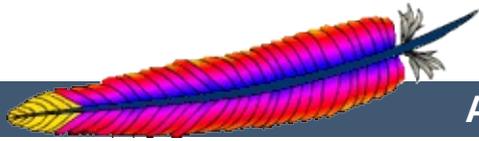
```

Premature end of script headers

```

CGI Internal Server Error
CGI

```



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



Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [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.



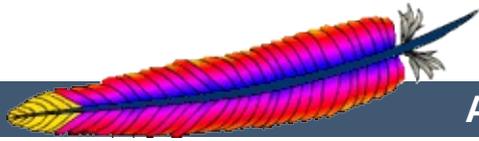
Please drop a note to dev@httpd.apache.org if you have additions or corrections to this list.

Library	Version	Thread Safe?	Notes
ASpell/PSpell		?	
Berkeley DB	3.x, 4.x	Yes	Be careful about sharing a connection across threads.
bzip2		Yes	Both low-level and high-level APIs are thread-safe. However, high-level API requires calling <code>errno</code> .
cdb		?	
C-Client		Perhaps	c-client uses <code>strtok()</code> and <code>gethostbyname()</code> which are not thread-safe on most implementations. c-client's static data is shared across threads. If <code>strtok()</code> and <code>gethostbyname()</code> are thread-safe, c-client <i>may</i> be thread-safe.
cpdflib		?	
libcrypt		?	
Expat		Yes	Need a separate parser instance per thread.
FreeTDS		?	
FreeType		?	
GD 1.8.x		?	
GD 2.0.x		?	
gdbm		No	Errors returned via a static <code>gdbm</code> pointer.
ImageMagick	5.2.2	Yes	ImageMagick docs claim it is thread-safe starting in version 5.2.2 (see Change log).
Imlib2		?	
libjpeg	v6b	?	

libmysqlclient		Yes	Use mysqlclient_r library variant for thread safety. For more information, please see http://www.mysql.com/doc/en/Thread-Safety.html
Ming	0.2a	?	
Net-SNMP	5.0.x	?	
OpenLDAP	2.1.x	Yes	Use ldap_r library variant to ensure thread safety
OpenSSL	0.9.6g	Yes	Requires proper usage of CRYPTO_THREAD_SUPPORT. See http://www.openssl.org/docs/crypto/CRYPTO_THREAD_SUPPORT.html for more information.
liboci8 (Oracle 8+)	8.x,9.x	?	
pdflib	5.0.x	Yes	PDFLib docs claim it is thread safe. However, the docs indicate it has been partially threaded since version V1.91: http://www.pdflib.com/products/pdflib.html
libpng	1.0.x	?	
libpng	1.2.x	?	
libpq (PostgreSQL)	7.x	Yes	Don't share connections across threads. Use <code>PGSQL_THREAD_SAFE</code> out for <code>crypt ()</code> calls
Sablotron	0.95	?	
zlib	1.1.4	Yes	Relies upon thread-safe <code>zalloc</code> and <code>zfree</code> . Default is to use libc's <code>calloc/free</code> which is not thread safe.

Copyright 2013 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 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

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](#)

[Technical details](#)

[Practical solutions to common problems](#)



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.



Regular Expressions

Basic regex building blocks



REWRITERULE BASICS

Basic anatomy of a RewriteRule, with exhaustively annotated simple examples.



Discussion of the flags to RewriteRule, and when and why one might use them.



REWRITE CONDITIONS

Discussion of RewriteCond, looping, and other related concepts.



RewriteMap

Discussion of RewriteMap, including simple, but heavily annotated, examples.



Discussion of the differences between rewrite rules in httpd.conf and in .htaccess files.



Environment Variables

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, i.e., before* any rewriting. This is important 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/
```

Copyright 2013 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](#).

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