

| | [FAQ](#) | |



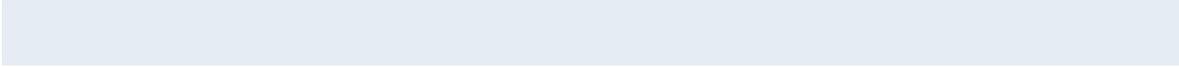
## Apache HTTP 2.0

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

# Apache HTTP 2.0



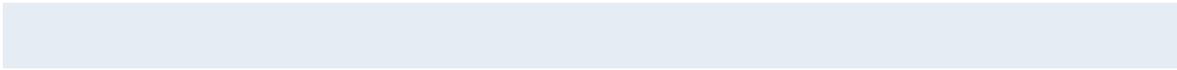
Google Search



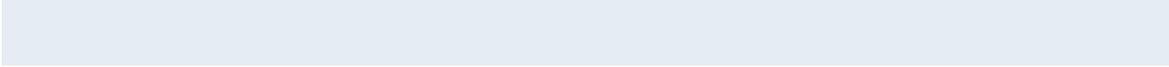
[Apache 2.0](#)

[1.3 2.0](#)

[Apache](#)



(MPM)



[\(DSO\)](#)

[URL](#)

[SSL/TLS](#)

[CGI Suexec](#)

[URL Rewriting](#)

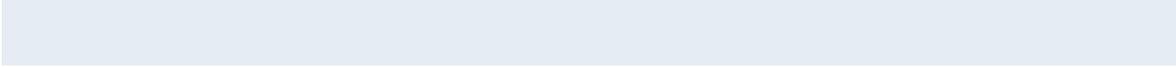
## How-To /

[CGI:](#)

[.htaccess](#)

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

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

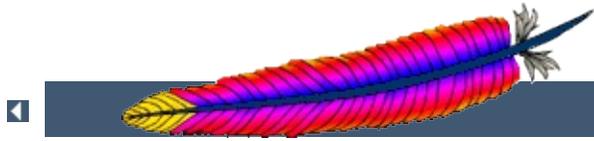


[Microsoft Windows](#)

[Novell NetWare](#)

[EBCDIC](#)

[\(FAQ\)](#)



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

## 1.3 2.0

Apache

[Apache 2.0](#)



- 
- Apache autoconf libtool Apac  
APACI
  - Apache 2.0



- Apache 1.3 MPM Apache
- Proxy HTTP/1.1
- <Proxy>
- `PATH_INFO ( )`
- PHP `PATH_INFO`
- `SSI` `PATH_INFO`
- CacheNegotiatedDocs `on off`
- `CacheNegotiatedDocs CacheNegotiatedDocs on`
- ErrorDocument

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

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

- `AccessConfig` `URL` `ResourceConfig`
- `httpd.conf`
- `Include conf/srm.conf` `Apache`
- `httpd.conf srm.confaccess.conf` I
- `BindAddress` `Port`
- `Port` `Apache-1.3` `URL`
- ServerName `L`
- `ServerName` `MPM`
- `AgentLog` `RefererLog` `RefererLog`
- `mod_log_agent` `mod_log_referer`
- `mod_log_config` CustomLog
- `AddModule` `ClearModuleList`

API

- FancyIndexing  
FancyIndexing
- mod\_negotiation MultiViews  
MultiviewsMatch
- (2.0.51 )  
ErrorHandler

Index0

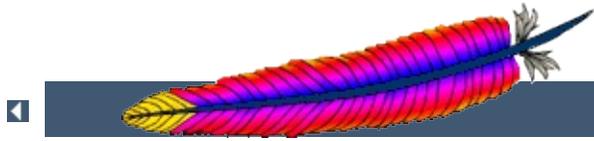
Header always set foo bar



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







| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache 2.0

Apache HTTP 1.3 2.0

[1.3 2.0](#)



---

## Unix

POSIX Unix

Apache

autoconf libtool

Apache conf

Apache

mod

## Unix

Apache 2.0 BeOS/OS/2/Windows Unix  
(MPM) Apache Portable Runtime (APR)  
POSIX

API

## Apache API

2.0 API

1.3 /

## IPv6

Apache Apache Portable Runtime library  
Apache IPv6 listen  
NameVirtualHost, VirtualHost IPv6  
"Listen [2001:db8::1]:8080")

IPv6

Listen

Apache  
Server Side Include

SSI

vhost

## Windows NT Unicode

Windows NT Apache 2.0  
Windows 2000 Windows XP  
*Windows 95, 98, ME*

Windows NT

utf-8

Apache 2.0 Perl (PCRE)

Pe



---

### mod\_ssl

Apache 2.0 OpenSSL

SSL/TLS

### mod\_dav

Apache 2.0

Versioning (DAV)

### mod\_deflate

Apache 2.0

### mod\_auth\_ldap

Apache 2.0.41 HTTP

L

### mod\_auth\_digest

### mod\_charset\_lite

Apache 2.0

### mod\_file\_cache

Apache 2.0

Apache 1.3

m

### mod\_headers

Apache 2.0

mod\_proxy

### mod\_proxy

proxy

HTTP/1.1 proxy

proxy ()

proxy\_connect, proxy\_ftp, proxy\_http

### mod\_negotiation

NOT ACCEPTABLE MULTIPLE CHOICES

ForceLanguagePriority

### mod\_autoindex

Autoindex

HTML

## mod\_include

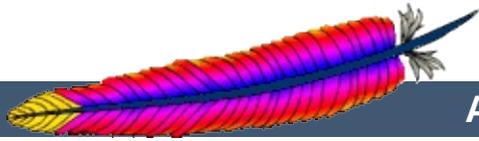
SSI

SSI

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.

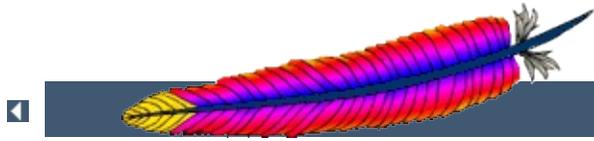
```
Copyright [yyyy] [name of copyright owner]
```

```
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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)



This translation may be out of date. Check the English version for recent changes.

Unix Unix

Apache Windows

[Apache](#)

Apache 2.0 Apache 1.3

Open Source

libtool autoc

(2.0.50 2.0.51 )

[Apache](#)

[Apache](#)



```
$ lynx http://httpd.apache.org/download.cgi
$ gzip -d httpd-2_0_NN.tar.gz
$ tar xvf httpd-2_0_NN.tar
$ ./configure --prefix=PREFIX
$ make
$ make install
$ vi PREFIX/conf/httpd.conf
$ PREFIX/bin/apachectl start
```

*NN*

*PREFIX*

/usr/local/apache2

Apache HTTPD



---

Apache :

50 MB

Apache 10 I

### ANSI-C

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

HTTP  
xntpd NTP NTP  
[comp.protocols.time.ntp](#) [NTP](#)

### Perl 5 []

[apxs](#) [dbmmanage](#) Perl Perl 5 (5.00  
"configure" Apache  
Perl 4 Perl 5 ) --with-perl () ./config



---

Apache [Apache HTTP](#)

Apache HTTP



---

Apache HTTPD tarball

tar :

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



---

```
autoconf libtool          Apache
                           buildconf

                           ./cor

Apache                    --prefix /

                           Apache
enable-module            module
enable-module=shared    (DSO)
disable-module          Base
```

```
configure
configure
```

```
/sw/pkg,
```

```
DSO :
```

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

```
configure
```

```
Makefile
```

```
configure
```

```
configure
```



Apache

:

```
$ make
```

Pentium III/Linux 2.2

3



---

*PREFIX* ( --prefix )

```
$ make install
```



*PREFIX/conf/*

Apache HTTP

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

[docs/manual/](#) Apache

<http://http>



---

Apache HTTP :

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

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

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



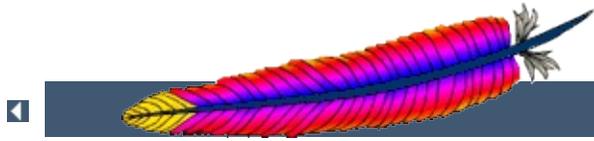
---

API

configure

( 2.0.55 2.0.57 )  
API  
(2.0.41

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache

This translation may be out of date. Check the English version for recent changes.

Windows Apache  
ME

Windows NT, 2000, XP Windows 9x,

Unix [httpd](#)

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



## Apache

```
Listen 80 (1024)
```

```
listen
```

```
httpd  
apachectl  
HTTPD
```

```
apachectl
```

```
httpd
```

```
httpd
```

```
httpd
```

```
httpd.conf
```

```
/usr/local/apache2/bin/apachectl -f  
/usr/local/apache2/conf/httpd.conf
```



---

Apache

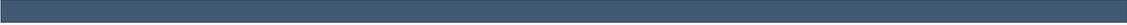
[Error](#)

...

- root
- Apache

Apache [FAQ](#)





apachec

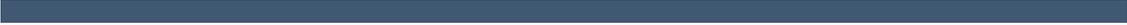
root

apachect1 SysV init

httpd

init



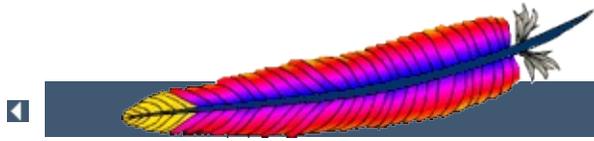


[httpd](#) [apachectl](#)

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)



This translation may be out of date. Check the English version for recent changes.

Unix  
Windows 9x, ME

Apache  
Apache

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



Apache

httpd

pid

[USR1](#)

:

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

httpd 2

-k

[httpd](#)

[apachectl](#)

[a](#)

httpd

:

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

[ServerRoot](#) [PidFile](#)



---

**: TERM**

apachectl -k stop

TERM stop kill



---

**: USR1**

apachectl -k graceful

USR1 graceful

USR1

( WI

MPM

mod\_status

USR1

status

USR1

([httpd](#) )  
httpd )



---

**: HUP**

apachectl -k restart

HUP restart

TERM kill

mod\_status

HUP

---



---

Apache 1.2b9 (:

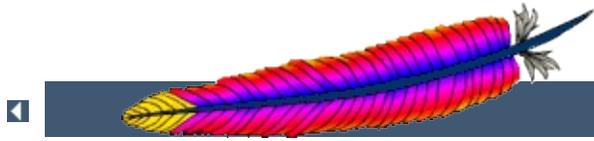
ScoreBoardFile

(HUP ) "long lost child came home!" (USR1 )

(HUP)

HTTP (KeepAlive)

KeepAlive



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)



This translation may be out of date. Check the English version for recent changes.

## Apache HTTP



```
mod_mime <IfDefine>  
    Include  
    TypesConfig
```

Apache  
Include

MIME



---

Apache 1 1

"\"

"#"

apachectl configtest



```
mod_so <IfModule>
        LoadModule
```

Apache

LoadModule

Apache

-1



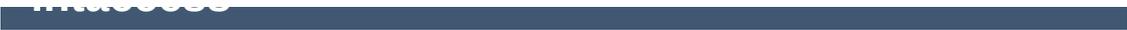
---

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

<FilesMatch>, <Location>, <LocationMatch>

Apache





```
AccessFileName  
AllowOverride
```

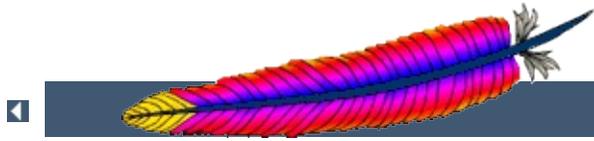
Apache

.htaccess

.htaccess

.htaccess

[.htaccess](#)



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

This translation may be out of date. Check the English version for recent changes.

URL `.htaccess`



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

<IfDefine>   <IfModule>

```
<IfDefine> httpd        httpd -DClosedForNow
:
```

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

<IfModule>            LoadModule

mod\_mime\_magic   MimeMagicFiles

```
<IfModule mod_mime_magic.c>
MimeMagicFile conf/magic
</IfModule>
```

<IfDefine> <IfModule> "!"



---

Apache Unix  
/usr/local/apache2 Windows "c:/Program  
Files/Apache Group/Apache2" (Apache Windows  
) Unix Apache  
/usr/local/apache2/htdocs/dir/

<Directory> <Files> <Directory>  
.htaccess /var/web/dir1

```
<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 Apache  
server-status

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

<Directory>, <Files>, <Location> C  
shell "\*" "?" 1 "[

(regex) <DirectoryMatch>, <FilesMatch>,  
<LocationMatch> perl regex

regex

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

regex

```
<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://yoursite.example.com/DIR/  
(  
Options)

<Location /> URL



---

<VirtualHost>



---

<Proxy>   <ProxyMatch>   URL   mod\_proxy  
cnn.com

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



Context <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>

```
<Directory /var/web/dir1> <Directory  
/var/web/dir/subdir> <Directory>  
Include Include
```

<VirtualHost>

```
(URL Alias  
<Location>/<LocationMatch>
```

A > B > C > D > E

```
<Location />  
E  
</Location>  
  
<Files f.html>  
D  
</Files>  
  
<VirtualHost *>  
<Directory /a/b>
```

```
B
</Directory>
</VirtualHost>

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

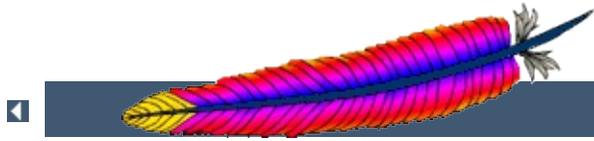
<Directory /a/b>
A
</Directory>
```

<Directory>

<Location>

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

# Woops! This <Directory> section will have no effect
<Directory />
Order allow,deny
Allow from all
Deny from badguy.example.com
</Directory>
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

core



```
ServerName  
ServerAdmin  
ServerSignature  
ServerTokens  
UseCanonicalName
```

ServerAdmin  
Server HTTP

ServerTokens

ServerName  
Apache

UseCanonicalName URL



---

CoreDumpDirectory

DocumentRoot

ErrorLog

LockFile

PidFile

ScoreBoardFile

ServerRoot

Apache



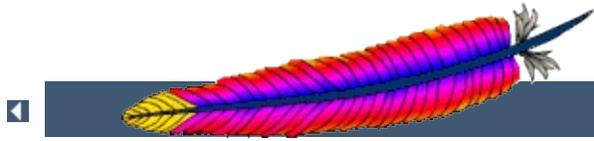
[LimitRequestBody](#)  
[LimitRequestFields](#)  
[LimitRequestFieldsize](#)  
[LimitRequestLine](#)  
[RLimitCPU](#)  
[RLimitMEM](#)  
[RLimitNPROC](#)  
[ThreadStackSize](#)

[LimitRequest\\*](#) Apache

[RLimit\\*](#) Apache

fork

[ThreadStackSize](#) Netware



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

--

This translation may be out of date. Check the English version for recent changes.



---

Apache

uid



```
ErrorLog
LogLevel
```

ErrorLog

( unix error\_logWin  
Unix syslog

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

CGI

```
tail -f error_log
```



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

Apache httpd mod\_log\_config,  
TransferLog

mod\_log\_agent

C printf(1)

## Common Log Format

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

common

"\t"

CustomLog

### Common Log Format (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

- (%1)

IdentityCheck On

frank (%u)

HTTP

ID CGI

401

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

:

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

day = 2\*digit

month = 3\*letter

year = 4\*digit

hour = 2\*digit

minute = 2\*digit

second = 2\*digit

zone = ('+' | '-' ) 4\*digit

%{format}t

"GET /apache\_pb.gif HTTP/1.0" ("%r")

HTTP/1.0

"%r"

200 (%>s)

(2)

)

2326 (%b)

## Combined Log Format

### Combined Log Format

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined
CustomLog log/access_log combined
```

### Common Log Format

HTTP

:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET
/apache_pb.gif HTTP/1.0" 200 2326
"http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98;
I ;Nav)"
```

:

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

    "Referer" () HTTP

)

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

    User-Agent HTTP

### CustomLog

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

:

```
# Mark requests from the loop-back interface
SetEnvIf Remote_Addr "127\.0\.0\.1" dontlog
# Mark requests for the robots.txt file
SetEnvIf Request_URI "^/robots\.txt$" dontlog
# Log what remains
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
```



---

open

10,000  
open

open

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



---

Apache httpd

( )

Apache httpd

ID

Apache HT

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

[cronolog](#)



---

>CustomLog ErrorLog

<VirtualHost>

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

%v



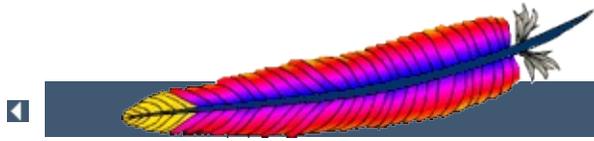
<u>mod cgi</u>	<u>PidFile</u>
<u>mod rewrite</u>	<u>RewriteLog</u>
	<u>RewriteLogLevel</u>
	<u>ScriptLog</u>
	<u>ScriptLogBuffer</u>
	<u>ScriptLogLength</u>

**PID**

Apache httpd ID logs/httpd.pid  
PidFile ID

ScriptLog CGI

mod rewrite  
RewriteLogLevel



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

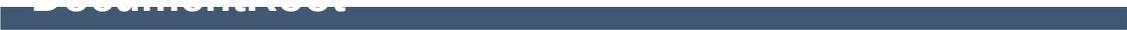
# URL

Apache 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>





Apache URL-Path

DocumentRoot

Apache  
IP



## DocumentRoot

### DocumentRoot

SymLinksIfOwnerMatch

### Alias

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

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

### AliasMatch

### ScriptAli

```
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



```
Unix    user          ~user/    mod_userdir
```

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

```
URL    /home/user/public_html/file.html  
       /home/user/    /etc/passwd
```

```
Userdir    /etc/passwd
```

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

```
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
```



Apache

DocumentRoot

/foo/ /bar/

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

/foo/ URL-Path  
/foo/

www.example.com /bar

Apache

RedirectMatch

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

:

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



---

Apache URL

`/foo/`      `internal.example.com`    `/bar/`

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

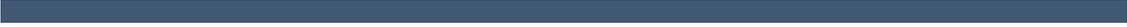
ProxyPass      ProxyPassReverse

`internal.example.com`

`internal.example.com`

`internal.example.com`

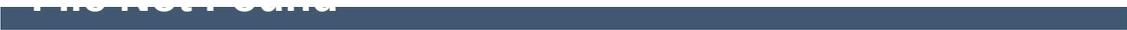




mod\_rewrite

() mod\_rewrite





URL

"File Not Found"  
mod\_speling ()  
Found"

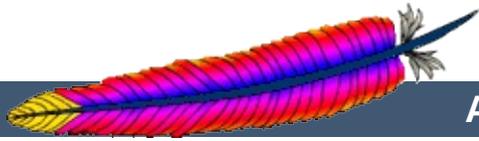
HTML URL  
(: spelling)

mod\_speling  
mod\_speling

URL unix

Apache HTTP 404

ErrorDocument



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



## Apache HTTP Server Version 2.0

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

## Security Tips

Some hints and tips on security issues in setting up a web server.  
Some of the suggestions will be general, others specific to Apache.



The Apache HTTP Server has a good record for security and a developer community highly concerned about security issues. But it is inevitable that some problems -- small or large -- will be discovered in software after it is released. For this reason, it is crucial to keep aware of updates to the software. If you have obtained your version of the HTTP Server directly from Apache, we highly recommend you subscribe to the [Apache HTTP Server Announcements List](#) where you can keep informed of new releases and security updates. Similar services are available from most third-party distributors of Apache software.

Of course, most times that a web server is compromised, it is not because of problems in the HTTP Server code. Rather, it comes from problems in add-on code, CGI scripts, or the underlying Operating System. You must therefore stay aware of problems and updates with all the software on your system.



In typical operation, Apache is started by the root user, and it switches to the user defined by the `User` directive to serve hits. As is the case with any command that root executes, you must take care that it is protected from modification by non-root users. Not only must the files themselves be writeable only by root, but so must the directories, and parents of all directories. For example, if you choose to place `ServerRoot` in `/usr/local/apache` then it is suggested that you create that directory as root, with commands like these:

```
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
```

It is assumed that `/`, `/usr`, and `/usr/local` are only modifiable by root. When you install the `httpd` executable, you should ensure that it is similarly protected:

```
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
```

You can create an `htdocs` subdirectory which is modifiable by other users -- since root never executes any files out of there, and shouldn't be creating files in there.

If you allow non-root users to modify any files that root either executes or writes on then you open your system to root compromises. For example, someone could replace the `httpd` binary so that the next time you start it, it will execute some arbitrary code. If the logs directory is writeable (by a non-root user), someone could replace a log file with a symlink to some

other system file, and then root might overwrite that file with arbitrary data. If the log files themselves are writeable (by a non-root user), then someone may be able to overwrite the log itself with bogus data.



---

Server Side Includes (SSI) present a server administrator with several potential security risks.

The first risk is the increased load on the server. All SSI-enabled files have to be parsed by Apache, whether or not there are any SSI directives included within the files. While this load increase is minor, in a shared server environment it can become significant.

SSI files also pose the same risks that are associated with CGI scripts in general. Using the `exec` cmd element, SSI-enabled files can execute any CGI script or program under the permissions of the user and group Apache runs as, as configured in `httpd.conf`.

There are ways to enhance the security of SSI files while still taking advantage of the benefits they provide.

To isolate the damage a wayward SSI file can cause, a server administrator can enable [suexec](#) as described in the [CGI in General](#) section.

Enabling SSI for files with `.html` or `.htm` extensions can be dangerous. This is especially true in a shared, or high traffic, server environment. SSI-enabled files should have a separate extension, such as the conventional `.shtml`. This helps keep server load at a minimum and allows for easier management of risk.

Another solution is to disable the ability to run scripts and programs from SSI pages. To do this replace `Includes` with `IncludesNOEXEC` in the [Options](#) directive. Note that users may still use `<--#include virtual="..." -->` to execute CGI scripts if these scripts are in directories designated by a [ScriptAlias](#) directive.



---

First of all, you always have to remember that you must trust the writers of the CGI scripts/programs or your ability to spot potential security holes in CGI, whether they were deliberate or accidental. CGI scripts can run essentially arbitrary commands on your system with the permissions of the web server user and can therefore be extremely dangerous if they are not carefully checked.

All the CGI scripts will run as the same user, so they have potential to conflict (accidentally or deliberately) with other scripts e.g. User A hates User B, so he writes a script to trash User B's CGI database. One program which can be used to allow scripts to run as different users is [suEXEC](#) which is included with Apache as of 1.2 and is called from special hooks in the Apache server code. Another popular way of doing this is with [CGIWrap](#).



Allowing users to execute CGI scripts in any directory should only be considered if:

- You trust your users not to write scripts which will deliberately or accidentally expose your system to an attack.
- You consider security at your site to be so feeble in other areas, as to make one more potential hole irrelevant.
- You have no users, and nobody ever visits your server.



## Script Aliased CGI

Limiting CGI to special directories gives the admin control over what goes into those directories. This is inevitably more secure than non script aliased CGI, but only if users with write access to the directories are trusted or the admin is willing to test each new CGI script/program for potential security holes.

Most sites choose this option over the non script aliased CGI approach.



Embedded scripting options which run as part of the server itself, such as `mod_php`, `mod_perl`, `mod_tcl`, and `mod_python`, run under the identity of the server itself (see the [User](#) directive), and therefore scripts executed by these engines potentially can access anything the server user can. Some scripting engines may provide restrictions, but it is better to be safe and assume not.



## Preventing .htaccess Overrides

To run a really tight ship, you'll want to stop users from setting up .htaccess files which can override security features you've configured. Here's one way to do it.

In the server configuration file, put

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

This prevents the use of .htaccess files in all directories apart from those specifically enabled.



## Prevent Default Access by Default

One aspect of Apache which is occasionally misunderstood is the feature of default access. That is, unless you take steps to change it, if the server can find its way to a file through normal URL mapping rules, it can serve it to clients.

For instance, consider the following example:

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

This would allow clients to walk through the entire filesystem. To work around this, add the following block to your server's configuration:

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

This will forbid default access to filesystem locations. Add appropriate [Directory](#) blocks to allow access only in those areas you wish. For example,

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

Pay particular attention to the interactions of [Location](#) and [Directory](#) directives; for instance, even if `<Directory />` denies access, a `<Location />` directive might overturn it.

Also be wary of playing games with the [UserDir](#) directive; setting

it to something like `./` would have the same effect, for root, as the first example above. If you are using Apache 1.3 or above, we strongly recommend that you include the following line in your server configuration files:

```
UserDir disabled root
```



## Monitoring Your Logs

To keep up-to-date with what is actually going on against your server you have to check the [Log Files](#). Even though the log files only reports what has already happened, they will give you some understanding of what attacks is thrown against the server and allow you to check if the necessary level of security is present.

A couple of examples:

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

The first example will list the number of attacks trying to exploit the [Apache Tomcat Source.JSP Malformed Request Information Disclosure Vulnerability](#), the second example will list the ten last denied clients, for example:

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

As you can see, the log files only report what already has happened, so if the client had been able to access the `.htpasswd` file you would have seen something similar to:

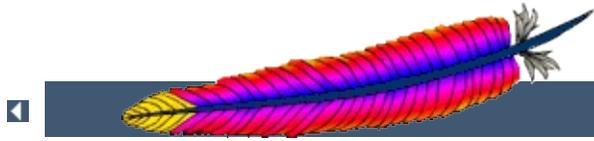
```
foo.bar.com - - [12/Jul/2002:01:59:13 +0200] "GET /.htpasswd  
HTTP/1.1"
```

in your [Access Log](#). This means you probably commented out the following in your server configuration file:

```
<Files ~ "^\.ht">  
Order allow,deny  
Deny from all  
</Files>
```

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

## (DSO)

This translation may be out of date. Check the English version for recent changes.

Apache HTTP  
Shared Object) (DSO)

DSO

DSO



```
mod_so LoadModule
```

Apache DSO  
core.c DSO  
enable-module=shared  
DSO

Apache () DSO  
Apache DSO  
Apache C DSO  
DSO

mod\_so.c  
Apache  
DSO DSO  
httpd.conf mod\_so

: Apache

Apache



## Apache 2.0 DSO

:

1. Apache mod\_foo.c DSO mod\_foo.so :

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

2. Apache mod\_foo.c DSO mod\_foo.so :

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

3. Apache :

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

4. Apache mod\_foo.c [apxs](#) Apache :

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

httpd.conf [Loa](#)



---

Unix OS (DSO) /

: 1

DSO DSO DSO libfoo.so 1:  
( /usr/lib)  
/usr/lib -R  
libfoo.so () DSO

DSO (DSO )  
)

DSO DSO ( )  
dlopen() DSO  
) DS

DSO API

DSO : DSO  
(

DSO

1998 DSO : Perl 5  
Apache  
Apache DSO



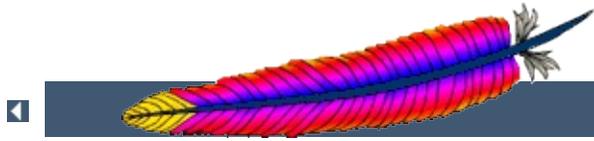
---

DSO :

- `configure`  
Apache ( SSL [mod\_perl, PHP3]
- Apache
- Apache DSO/ apxs Apache  
`apxs -i apachectl restart`

DSO :

- 
- Unix 20%
- (PIC) ( position independent code)
- DSO DSO ( DS  
) DSO PI  
Apache  
`dlopen ()`



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

This translation may be out of date. Check the English version for recent changes.

Apache HTTP/1.1

mod\_negotiation



---

Accept-Language: fr

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

Apache HTTP/1.1 'server driven'  
Language, Accept -Charset, Accept -Encoding Apache  
'transparent' RFC 2295  
'feature negotiation'

URI (RFC 2396) Apache HTTP  
0 1



variant

- ( \*.var ) variant
- 'Multiviews'

### type-map

type-map ( Apache MIME  
 type-map)

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

) foo variant foo.1

```
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
```

### MultiViews

"qs"

```
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 variant    'qs'  
variant qs 1.0 qs         variant  
JPEG                      ASCII  
qs      variant
```

### [mod\\_negotiation](#)

## Multiviews

```
MultiViews                  httpd.conf      <Dir  
<Files> (                  AllowOverride )  
Options All MultiViews
```

```
MultiViews:                /some/dir/foo  
MultiViews                  /some/dir/foo
```

```
MultiViews Directo
```

```
DirectoryIndex index
```

```
index.html index.html3
```

MultiViews



---

## Apache variant

1. **Apache Server** driven negotiation  
Apache Apa
2. RFC 2295 trans  
variant  
2296 'remote variant selection algorithm'

Accept  
Accept-Language  
  
Accept-Encoding  
Accept-Charset

## Apache

variant ()

Apache

1. *Accept\**  
variant 4
2. variant var  
variant  
  1. variant Accept variant
  2. variant
  3. () Accept-Language ()  
LanguagePriority variant



Apache  
Accept

Apache

Accept:

"image/\*"

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

"image/"

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

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

1.0 ()

Accept: q  
"type/\*" 0.02 q

Apache "\*/\*" 0.01 q  
( "\*/\*" )

Apache 2.0

Acc

"Multiple Choices"

LanguagePriority

Language	en-GB	en
Acceptable Variants"		<u>LanguagePriority</u>
en	Apache	
"fr"		"fr"

(Cookie URL )

mod\_negotiation prefer-language  
mod\_negotiation variant

### Example

```
SetEnvIf Cookie "language=en" prefer-language=en  
SetEnvIf Cookie "language=fr" prefer-language=fr
```



## Transparent Content Negotiation

Apache transparent content negotiation (RFC 2295)  
variant {encoding ..} variant variant  
Accept-Encoding variant variant RVSA/1.0  
(RFC 2296) RVSA/1.0 variant 5



---

MIME ( html) ( gz)

:

- 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)

MIME ( foo.html) ()



---

URL

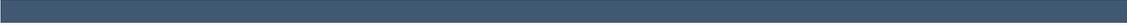
HTTP/1.0

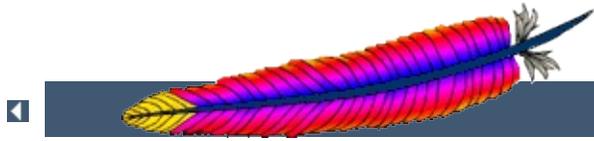
HTTP/1.0

HTTP/1.1

()







| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)



Apache

"500 Server Error"



## NCSA httpd 1.3 /

1. NCSA
2. URL
3. URL

URL /

Apache 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 URL (CGI  
CGI)



AllowOverride

.htaccess

ErrorD

```
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



---

/SSI

URL Apache

CGI

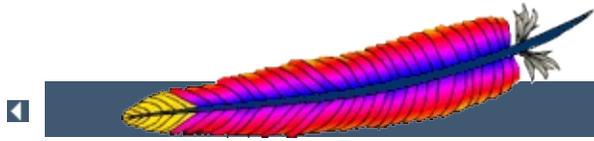
HTTP\_USER\_AGENT REDIRECT\_HTTP\_USER\_  
Apache REDIRECT\_  
URL

ErrorDocument CGI

ErrorDocument Perl

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

404 Not Found



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

This translation may be out of date. Check the English version for recent changes.

Apache

[DNS](#)



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

Apache  
IP Apache

### Listen

Listen

listen

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 IPv4 IPv6  
IPv6  
Apache

Apache

IPv4 IPv6  
configure

Listen

IPv4 IPv6

```
Listen 80
```

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

APR

IPv4

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

IPv4 IPv6  
configure

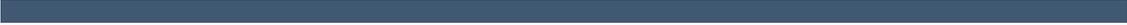
Listen

( IPv4 )

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

--disable-v4-mapped Apache  
disable-v4-mapped FreeBSD, NetBSD, OpenBSD





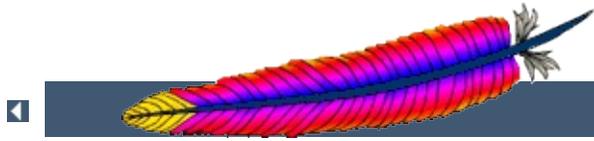
Listen

<VirtualHost>

<VirtualHost>

Listen listen

<VirtualHost> listen



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

## (MPM)

This translation may be out of date. Check the English version for recent changes.

Apache HTTP



---

Apache HTTP

Apache 2.0

:

- Apache  
Apache 1.3 POSIX

- 

([perchild](#))

MPM Apache

MPM



---

MPM

MPM Apache

MPM ./configure

--with-mpm= NA

MPM

MPM

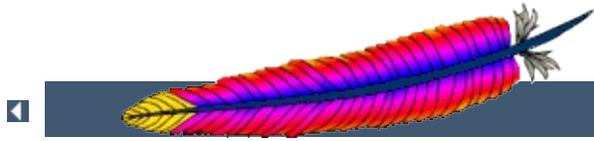
./httpd -l



---

## OS MPM MPM

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache

This translation may be out of date. Check the English version for recent changes.

Apache HTTP



<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>

Apache

```

mod_setenvif
referrer HTTP Referer
RewriteRule
)
[E=...]

```

mod\_unique\_id

**CGI**

Apache CGI SSI

- CGI
- CGI [suexec](#) CGI
- (:'\_')



<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 Apache

## SSI

mod\_include INCLUDES

server-parsed (SSI)

allow from env= deny from env=

LogFormat %e  
gif

Header

ExtFilterDefine  
enableenv=

mod\_ext\_filter

## URL

RewriteCond

%{ENV:...} mod\_rewrite

ENV:



---

Apache

[PassEnv](#)

**downgrade-1.0**

HTTP/1.0

HTTP/1.0

**force-no-vary**

Vary

**force-response-1.0**

HTTP/1.0

f

HTTP/1.1

**gzip-only-text/html**

1 text/html

[mod\\_deflate](#)

**no-gzip**

[mod\\_deflate](#) DEFLATE

**nokeepalive**

[KeepAlive](#)

**prefer-language**

[mod\\_negotiation](#)

variant

(en, ja, x-klingon)

**redirect-carefully**

## **suppress-error-charset**

*Apache 2.0.40*

0



## httpd.conf

```
#
# The following directives modify normal HTTP response behavior.
# The first directive disables keepalive for Netscape 2.x and bro
# spoof it. There are known problems with these browser implement
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properl
# support keepalive when it is used on 301 or 302 (redirect) resp
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-respo

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

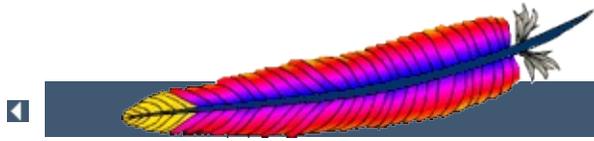
```
SetEnvIf Request_URI \.gif image-request
SetEnvIf Request_URI \.jpg image-request
SetEnvIf Request_URI \.png image-request
CustomLog logs/access_log common env=!image-request
```

## inline

```
SetEnvIf Referer "^http://www.example.com/" local_referal
# Allow browsers that do not send Referer info
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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache

Apache



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

Apache

Apache 1.1

Action

- **default-handler:** default\_handler()
- **send-as-is:** HTTP (mod\_asis)
- **cgi-script:** CGI (mod\_cgi)
- **imap-file:** (mod\_imap)
- **server-info:** (mod\_info)
- **server-status:** (mod\_status)
- **type-map:**



---

## CGI

html

footer.pl CGI

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

## CGI

(

PATH\_

## HTTP

send-as-is HTTP

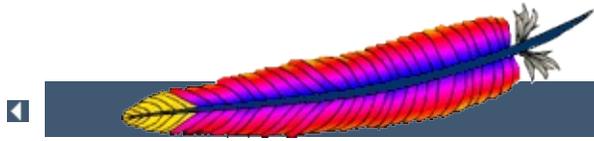
send-as-is

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



```
char *handler
```

```
(: "-")
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

Apache



<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>

Apache ()

SetOutputFilter, AddInputFilter, AddOutputFilter,  
RemoveInputFilter, RemoveOutputFilter

Apache HTTP

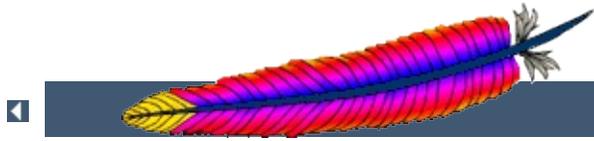
## INCLUDES

mod\_include Server-Side Include

## DEFLATE

mod\_deflate

mod\_ext\_filter



| [FAQ](#) |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# suEXEC

This translation may be out of date. Check the English version for recent changes.

suEXEC Apache Web ID  
SSI web

II

CGI  
suEXEC

SSI



---

Apache

1        **setuid**   **setgid**   UNIX

2

3        suEXEC suEXEC

Apache suEXEC

4 suEXEC Apache

suEXEC suEXEC

suEXEC

?!



suEXEC

**suEXEC** Apache web      setuid "wrapper"  
HTTP wrapper      Apache  
wrapper

wrapper

1. **wrapper ?**

wrapper

2. **wrapper ?**

wrapper  
suEXEC

Apac

3. **wrapper**

?

wrapper ?

(Apache )

4. **CGI, SSI ?**

CGI, SSI '/'  
-with-suexec-docroot=*DIR* )

'.' ?

5. ?

?

6. ?

?

7. ?

suEXEC

*root* CGI/SSI

8. **ID ID** ?  
ID CGI/SSI ID
9. ?  
suEXEC 'root' CGI/SSI
10. **ID ID** ?  
ID CGI/SSI
11. **wrapper ?**  
setuid setgid
12. **CGI/SSI** (change directory
13. **Apache ?**  
suEXEC ?  
([suEXEC](#) )
14. ?
15. **CGI/SSI ?**
16. **CGI/SSI**  
CGI/SSI
17. **CGI/SSI setuid setgid** ?

UID/GID

18. / *I?*

?

19. ?

suEXEC ()

20. **CGI/SSI exec ?**

suEXEC

suEXEC wrapper

suEXEC



## suEXEC

**--enable-suexec**

enable-suexec

suEXEC

--with-suexec

**--with-suexec-bin=PATH**

suexec

bin=/usr/sbin/suexec

**--with-suexec-caller=UID**

Apache suexec

**--with-suexec-userdir=DIR**

suEXEC

("\*")

Userdir

UserDir

**--with-suexec-docroot=DIR**

Apache suEXEC

"/htdocs"

wrapper "/home/apache/htdocs"

(UserDir

"--datadir=/home/a

**--with-suexec-uidmin=UID**

suEXEC UID

500 10

**--with-suexec-gidmin=GID**

suEXEC GID

100

**--with-suexec-logfile=FILE**

suEXEC

logfiledir)

()

**--with-suexec-safepath=PATH**

CGI PATH

"/usr/local/bin:/u

## suEXEC wrapper

```
--enable-suexec suEXEC
```

```
"make"
```

(Apache )

```
make install
```

```
"/usr/local/apache/sbin/suexec"
```

**root** wrapper ID

```
suEXEC
```

```
--with-suexec-caller configure
```

```
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
```

Apache

suEXEC



```
Apache --sbindir suexec
"/usr/local/apache/sbin/suexec") suEXEC
```

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

wrapper

```
suEXEC Apache Apache
```

```
suEXEC suexec Apache kill
```



CGI suEXEC

SuexecUserG

:

suEXEC wrapper

VirtualHost

SuexecUserG

ID CGI

<Virtual

>VirtualHost>

ID

:

mod\_userdir

ID CGI

--with-suexec-userdir



suEXEC wrapper      --with-suexec-logfile



---

!

Apache

wrapper

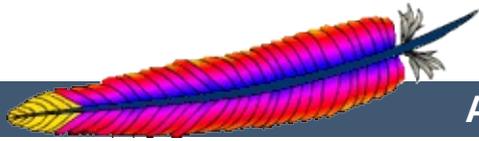
suEXEC ""

- **suEXEC**
- 

suEXEC

Apache

- suEXEC PATH
  
- suEXEC



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



## Apache HTTP Server Version 2.0

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

## Apache Performance Tuning

Apache 2.x is a general-purpose webserver, designed to provide a balance of flexibility, portability, and performance. Although it has not been designed specifically to set benchmark records, Apache 2.x is capable of high performance in many real-world situations.

Compared to Apache 1.3, release 2.x contains many additional optimizations to increase throughput and scalability. Most of these improvements are enabled by default. However, there are compile-time and run-time configuration choices that can significantly affect performance. This document describes the options that a server administrator can configure to tune the performance of an Apache 2.x installation. Some of these configuration options enable the httpd to better take advantage of the capabilities of the hardware and OS, while others allow the administrator to trade functionality for speed.



The single biggest hardware issue affecting webserver performance is RAM. A webserver should never ever have to swap, as swapping increases the latency of each request beyond a point that users consider "fast enough". This causes users to hit stop and reload, further increasing the load. You can, and should, control the `MaxClients` setting so that your server does not spawn so many children it starts swapping. This procedure for doing this is simple: determine the size of your average Apache process, by looking at your process list via a tool such as `top`, and divide this into your total available memory, leaving some room for other processes.

Beyond that the rest is mundane: get a fast enough CPU, a fast enough network card, and fast enough disks, where "fast enough" is something that needs to be determined by experimentation.

Operating system choice is largely a matter of local concerns. But some guidelines that have proven generally useful are:

- Run the latest stable release and patchlevel of the operating system that you choose. Many OS suppliers have introduced significant performance improvements to their TCP stacks and thread libraries in recent years.
- If your OS supports a `sendfile(2)` system call, make sure you install the release and/or patches needed to enable it. (With Linux, for example, this means using Linux 2.4 or later. For early releases of Solaris 8, you may need to apply a patch.) On systems where it is available, `sendfile` enables Apache 2 to deliver static content faster and with lower CPU utilization.



Related Modules	Related Directives
<a href="#">mod_dir</a>	<a href="#">AllowOverride</a>
<a href="#">mpm_common</a>	<a href="#">DirectoryIndex</a>
<a href="#">mod_status</a>	<a href="#">HostnameLookups</a>
	<a href="#">EnableMMAP</a>
	<a href="#">EnableSendfile</a>
	<a href="#">KeepAliveTimeout</a>
	<a href="#">MaxSpareServers</a>
	<a href="#">MinSpareServers</a>
	<a href="#">Options</a>
	<a href="#">StartServers</a>

### HostnameLookups and other DNS considerations

Prior to Apache 1.3, [HostnameLookups](#) defaulted to On. This adds latency to every request because it requires a DNS lookup to complete before the request is finished. In Apache 1.3 this setting defaults to Off. If you need to have addresses in your log files resolved to hostnames, use the [logresolve](#) program that comes with Apache, or one of the numerous log reporting packages which are available.

It is recommended that you do this sort of postprocessing of your log files on some machine other than the production web server machine, in order that this activity not adversely affect server performance.

If you use any [Allow](#) from domain or [Deny](#) from domain directives (i.e., using a hostname, or a domain name, rather than an IP address) then you will pay for two DNS lookups (a reverse, followed by a forward lookup to make sure that the reverse is not being spoofed). For best performance, therefore, use IP

addresses, rather than names, when using these directives, if possible.

Note that it's possible to scope the directives, such as within a `<Location /server-status>` section. In this case the DNS lookups are only performed on requests matching the criteria. Here's an example which disables lookups except for `.html` and `.cgi` files:

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

But even still, if you just need DNS names in some CGIs you could consider doing the `gethostbyname` call in the specific CGIs that need it.

## FollowSymLinks and SymLinksIfOwnerMatch

Wherever in your URL-space you do not have an `Options FollowSymLinks`, or you do have an `Options SymLinksIfOwnerMatch` Apache will have to issue extra system calls to check up on symlinks. One extra call per filename component. For example, if you had:

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

and a request is made for the URI `/index.html`. Then Apache will perform `lstat(2)` on `/www`, `/www/htdocs`, and `/www/htdocs/index.html`. The results of these `lstats` are never cached, so they will occur on every single request. If you really desire the symlinks security checking you can do something

like this:

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

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

This at least avoids the extra checks for the [DocumentRoot](#) path. Note that you'll need to add similar sections if you have any [Alias](#) or [RewriteRule](#) paths outside of your document root. For highest performance, and no symlink protection, set `FollowSymLinks` everywhere, and never set `SymLinksIfOwnerMatch`.

## AllowOverride

Wherever in your URL-space you allow overrides (typically `.htaccess` files) Apache will attempt to open `.htaccess` for each filename component. For example,

```
DocumentRoot /www/htdocs
<Directory />
  AllowOverride all
</Directory>
```

and a request is made for the URI `/index.html`. Then Apache will attempt to open `/.htaccess`, `/www/.htaccess`, and `/www/htdocs/.htaccess`. The solutions are similar to the previous case of `Options FollowSymLinks`. For highest performance use `AllowOverride None` everywhere in your filesystem.

## Negotiation

If at all possible, avoid content-negotiation if you're really interested in every last ounce of performance. In practice the benefits of negotiation outweigh the performance penalties. There's one case where you can speed up the server. Instead of using a wildcard such as:

```
DirectoryIndex index
```

Use a complete list of options:

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

where you list the most common choice first.

Also note that explicitly creating a `type-map` file provides better performance than using `MultiViews`, as the necessary information can be determined by reading this single file, rather than having to scan the directory for files.

If your site needs content negotiation consider using `type-map` files, rather than the `Options MultiViews` directive to accomplish the negotiation. See the [Content Negotiation](#) documentation for a full discussion of the methods of negotiation, and instructions for creating `type-map` files.

## Memory-mapping

In situations where Apache 2.x needs to look at the contents of a file being delivered--for example, when doing server-side-include processing--it normally memory-maps the file if the OS supports some form of `mmap(2)`.

On some platforms, this memory-mapping improves performance. However, there are cases where memory-mapping can hurt the performance or even the stability of the `httpd`:

- On some operating systems, `mmap` does not scale as well as `read(2)` when the number of CPUs increases. On multiprocessor Solaris servers, for example, Apache 2.x sometimes delivers server-parsed files faster when `mmap` is disabled.
- If you memory-map a file located on an NFS-mounted filesystem and a process on another NFS client machine deletes or truncates the file, your process may get a bus error the next time it tries to access the mapped file content.

For installations where either of these factors applies, you should use `EnableMMAP off` to disable the memory-mapping of delivered files. (Note: This directive can be overridden on a per-directory basis.)

## Sendfile

In situations where Apache 2.x can ignore the contents of the file to be delivered -- for example, when serving static file content -- it normally uses the kernel `sendfile` support the file if the OS supports the `sendfile(2)` operation.

On most platforms, using `sendfile` improves performance by eliminating separate read and send mechanics. However, there are cases where using `sendfile` can harm the stability of the `httpd`:

- 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.
- With an NFS-mounted files, the kernel may be unable to reliably serve the network file through it's own cache.

For installations where either of these factors applies, you should

use `EnableSendfile off` to disable sendfile delivery of file contents. (Note: This directive can be overridden on a per-directory basis.)

## Process Creation

Prior to Apache 1.3 the [MinSpareServers](#), [MaxSpareServers](#), and [StartServers](#) settings all had drastic effects on benchmark results. In particular, Apache required a "ramp-up" period in order to reach a number of children sufficient to serve the load being applied. After the initial spawning of [StartServers](#) children, only one child per second would be created to satisfy the [MinSpareServers](#) setting. So a server being accessed by 100 simultaneous clients, using the default [StartServers](#) of 5 would take on the order 95 seconds to spawn enough children to handle the load. This works fine in practice on real-life servers, because they aren't restarted frequently. But does really poorly on benchmarks which might only run for ten minutes.

The one-per-second rule was implemented in an effort to avoid swamping the machine with the startup of new children. If the machine is busy spawning children it can't service requests. But it has such a drastic effect on the perceived performance of Apache that it had to be replaced. As of Apache 1.3, the code will relax the one-per-second rule. It will spawn one, wait a second, then spawn two, wait a second, then spawn four, and it will continue exponentially until it is spawning 32 children per second. It will stop whenever it satisfies the [MinSpareServers](#) setting.

This appears to be responsive enough that it's almost unnecessary to twiddle the [MinSpareServers](#), [MaxSpareServers](#) and [StartServers](#) knobs. When more than 4 children are spawned per second, a message will be emitted to the [ErrorLog](#). If you see a lot of these errors then consider

tuning these settings. Use the [mod\\_status](#) output as a guide.

Related to process creation is process death induced by the [MaxRequestsPerChild](#) setting. By default this is 0, which means that there is no limit to the number of requests handled per child. If your configuration currently has this set to some very low number, such as 30, you may want to bump this up significantly. If you are running SunOS or an old version of Solaris, limit this to 10000 or so because of memory leaks.

When keep-alives are in use, children will be kept busy doing nothing waiting for more requests on the already open connection. The default [KeepAliveTimeout](#) of 15 seconds attempts to minimize this effect. The tradeoff here is between network bandwidth and server resources. In no event should you raise this above about 60 seconds, as [most of the benefits are lost](#).



## Choosing an MPM

Apache 2.x supports pluggable concurrency models, called [Multi-Processing Modules](#) (MPMs). When building Apache, you must choose an MPM to use. There are platform-specific MPMs for some platforms: [beos](#), [mpm\\_network](#), [mpm\\_os2](#), and [mpm\\_winnt](#). For general Unix-type systems, there are several MPMs from which to choose. The choice of MPM can affect the speed and scalability of the httpd:

- The [worker](#) MPM uses multiple child processes with many threads each. Each thread handles one connection at a time. Worker generally is a good choice for high-traffic servers because it has a smaller memory footprint than the prefork MPM.
- The [prefork](#) MPM uses multiple child processes with one thread each. Each process handles one connection at a time. On many systems, prefork is comparable in speed to worker, but it uses more memory. Prefork's threadless design has advantages over worker in some situations: it can be used with non-thread-safe third-party modules, and it is easier to debug on platforms with poor thread debugging support.

For more information on these and other MPMs, please see the MPM [documentation](#).

## Modules

Since memory usage is such an important consideration in performance, you should attempt to eliminate modules that you are not actually using. If you have built the modules as [DSOs](#), eliminating modules is a simple matter of commenting out the associated [LoadModule](#) directive for that module. This allows you to experiment with removing modules, and seeing if your site still

functions in their absence.

If, on the other hand, you have modules statically linked into your Apache binary, you will need to recompile Apache in order to remove unwanted modules.

An associated question that arises here is, of course, what modules you need, and which ones you don't. The answer here will, of course, vary from one web site to another. However, the *minimal* list of modules which you can get by with tends to include [mod\\_mime](#), [mod\\_dir](#), and [mod\\_log\\_config](#). `mod_log_config` is, of course, optional, as you can run a web site without log files. This is, however, not recommended.

## Atomic Operations

Some modules, such as [mod\\_cache](#) and recent development builds of the worker MPM, use APR's atomic API. This API provides atomic operations that can be used for lightweight thread synchronization.

By default, APR implements these operations using the most efficient mechanism available on each target OS/CPU platform. Many modern CPUs, for example, have an instruction that does an atomic compare-and-swap (CAS) operation in hardware. On some platforms, however, APR defaults to a slower, mutex-based implementation of the atomic API in order to ensure compatibility with older CPU models that lack such instructions. If you are building Apache for one of these platforms, and you plan to run only on newer CPUs, you can select a faster atomic implementation at build time by configuring Apache with the `--enable-nonportable-atomics` option:

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

The `--enable-nonportable-atomics` option is relevant for the following platforms:

- **Solaris on SPARC**  
By default, APR uses mutex-based atomics on Solaris/SPARC. If you configure with `--enable-nonportable-atomics`, however, APR generates code that uses a SPARC v8plus opcode for fast hardware compare-and-swap. If you configure Apache with this option, the atomic operations will be more efficient (allowing for lower CPU utilization and higher concurrency), but the resulting executable will run only on UltraSPARC chips.
- **Linux on x86**  
By default, APR uses mutex-based atomics on Linux. If you configure with `--enable-nonportable-atomics`, however, APR generates code that uses a 486 opcode for fast hardware compare-and-swap. This will result in more efficient atomic operations, but the resulting executable will run only on 486 and later chips (and not on 386).

## **mod\_status and ExtendedStatus On**

If you include `mod_status` and you also set `ExtendedStatus On` when building and running Apache, then on every request Apache will perform two calls to `gettimeofday(2)` (or `times(2)` depending on your operating system), and (pre-1.3) several extra calls to `time(2)`. This is all done so that the status report contains timing indications. For highest performance, set `ExtendedStatus off` (which is the default).

## **accept Serialization - multiple sockets**

**Warning:**

This section has not been fully updated to take into account changes made in the 2.x version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

This discusses a shortcoming in the Unix socket API. Suppose your web server uses multiple `Listen` statements to listen on either multiple ports or multiple addresses. In order to test each socket to see if a connection is ready Apache uses `select(2)`. `select(2)` indicates that a socket has *zero or at least one* connection waiting on it. Apache's model includes multiple children, and all the idle ones test for new connections at the same time. A naive implementation looks something like this (these examples do not match the code, they're contrived for pedagogical purposes):

```
for (;;) {
    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;
    }
    process the new_connection;
}
```

But this naive implementation has a serious starvation problem.

Recall that multiple children execute this loop at the same time, and so multiple children will block at `select` when they are in between requests. All those blocked children will awaken and return from `select` when a single request appears on any socket (the number of children which awaken varies depending on the operating system and timing issues). They will all then fall down into the loop and try to accept the connection. But only one will succeed (assuming there's still only one connection ready), the rest will be *blocked* in `accept`. This effectively locks those children into serving requests from that one socket and no other sockets, and they'll be stuck there until enough new requests appear on that socket to wake them all up. This starvation problem was first documented in [PR#467](#). There are at least two solutions.

One solution is to make the sockets non-blocking. In this case the `accept` won't block the children, and they will be allowed to continue immediately. But this wastes CPU time. Suppose you have ten idle children in `select`, and one connection arrives. Then nine of those children will wake up, try to accept the connection, fail, and loop back into `select`, accomplishing nothing. Meanwhile none of those children are servicing requests that occurred on other sockets until they get back up to the `select` again. Overall this solution does not seem very fruitful unless you have as many idle CPUs (in a multiprocessor box) as you have idle children, not a very likely situation.

Another solution, the one used by Apache, is to serialize entry into the inner loop. The loop looks like this (differences highlighted):

```
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;
}

```

The functions `accept_mutex_on` and `accept_mutex_off` implement a mutual exclusion semaphore. Only one child can have the mutex at any time. There are several choices for implementing these mutexes. The choice is defined in `src/conf.h` (pre-1.3) or `src/include/ap_config.h` (1.3 or later). Some architectures do not have any locking choice made, on these architectures it is unsafe to use multiple [Listen](#) directives.

The directive [AcceptMutex](#) can be used to change the selected mutex implementation at run-time.

### **AcceptMutex flock**

This method uses the `flock(2)` system call to lock a lock file (located by the [LockFile](#) directive).

### **AcceptMutex fcntl**

This method uses the `fcntl(2)` system call to lock a lock file (located by the [LockFile](#) directive).

### **AcceptMutex sysvsem**

(1.3 or later) This method uses SysV-style semaphores to

implement the mutex. Unfortunately SysV-style semaphores have some bad side-effects. One is that it's possible Apache will die without cleaning up the semaphore (see the `ipcs(8)` man page). The other is that the semaphore API allows for a denial of service attack by any CGIs running under the same uid as the webserver (*i.e.*, all CGIs, unless you use something like `suexec` or `cgiwrapper`). For these reasons this method is not used on any architecture except IRIX (where the previous two are prohibitively expensive on most IRIX boxes).

### **AcceptMutex pthread**

(1.3 or later) This method uses POSIX mutexes and should work on any architecture implementing the full POSIX threads specification, however appears to only work on Solaris (2.5 or later), and even then only in certain configurations. If you experiment with this you should watch out for your server hanging and not responding. Static content only servers may work just fine.

### **AcceptMutex posixsem**

(2.0 or later) This method uses POSIX semaphores. The semaphore ownership is not recovered if a thread in the process holding the mutex segfaults, resulting in a hang of the web server.

If your system has another method of serialization which isn't in the above list then it may be worthwhile adding code for it to APR.

Another solution that has been considered but never implemented is to partially serialize the loop -- that is, let in a certain number of processes. This would only be of interest on multiprocessor boxes where it's possible multiple children could run simultaneously, and the serialization actually doesn't take advantage of the full bandwidth. This is a possible area of future investigation, but priority remains low because highly parallel web servers are not

the norm.

Ideally you should run servers without multiple [Listen](#) statements if you want the highest performance. But read on.

## **accept Serialization - single socket**

The above is fine and dandy for multiple socket servers, but what about single socket servers? In theory they shouldn't experience any of these same problems because all children can just block in `accept(2)` until a connection arrives, and no starvation results. In practice this hides almost the same "spinning" behaviour discussed above in the non-blocking solution. The way that most TCP stacks are implemented, the kernel actually wakes up all processes blocked in `accept` when a single connection arrives. One of those processes gets the connection and returns to user-space, the rest spin in the kernel and go back to sleep when they discover there's no connection for them. This spinning is hidden from the user-land code, but it's there nonetheless. This can result in the same load-spiking wasteful behaviour that a non-blocking solution to the multiple sockets case can.

For this reason we have found that many architectures behave more "nicely" if we serialize even the single socket case. So this is actually the default in almost all cases. Crude experiments under Linux (2.0.30 on a dual Pentium pro 166 w/128Mb RAM) have shown that the serialization of the single socket case causes less than a 3% decrease in requests per second over unserialized single-socket. But unserialized single-socket showed an extra 100ms latency on each request. This latency is probably a wash on long haul lines, and only an issue on LANs. If you want to override the single socket serialization you can define `SINGLE_LISTEN_UNSERIALIZED_ACCEPT` and then single-socket servers will not serialize at all.

## Lingering Close

As discussed in [draft-ietf-http-connection-00.txt](#) section 8, in order for an HTTP server to **reliably** implement the protocol it needs to shutdown each direction of the communication independently (recall that a TCP connection is bi-directional, each half is independent of the other). This fact is often overlooked by other servers, but is correctly implemented in Apache as of 1.2.

When this feature was added to Apache it caused a flurry of problems on various versions of Unix because of a shortsightedness. The TCP specification does not state that the FIN\_WAIT\_2 state has a timeout, but it doesn't prohibit it. On systems without the timeout, Apache 1.2 induces many sockets stuck forever in the FIN\_WAIT\_2 state. In many cases this can be avoided by simply upgrading to the latest TCP/IP patches supplied by the vendor. In cases where the vendor has never released patches (*i.e.*, SunOS4 -- although folks with a source license can patch it themselves) we have decided to disable this feature.

There are two ways of accomplishing this. One is the socket option SO\_LINGER. But as fate would have it, this has never been implemented properly in most TCP/IP stacks. Even on those stacks with a proper implementation (*i.e.*, Linux 2.0.31) this method proves to be more expensive (cputime) than the next solution.

For the most part, Apache implements this in a function called `lingering_close` (in `http_main.c`). The function looks roughly like this:

```
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);
}

```

This naturally adds some expense at the end of a connection, but it is required for a reliable implementation. As HTTP/1.1 becomes more prevalent, and all connections are persistent, this expense will be amortized over more requests. If you want to play with fire and disable this feature you can define `NO_LINGCLOSE`, but this is not recommended at all. In particular, as HTTP/1.1 pipelined persistent connections come into use `lingering_close` is an absolute necessity (and [pipelined connections are faster](#), so you want to support them).

## Scoreboard File

Apache's parent and children communicate with each other through something called the scoreboard. Ideally this should be implemented in shared memory. For those operating systems that we either have access to, or have been given detailed ports for, it typically is implemented using shared memory. The rest default to using an on-disk file. The on-disk file is not only slow, but it is unreliable (and less featured). Peruse the `src/main/conf.h` file for your architecture and look for either `USE_MMAP_SCOREBOARD` or `USE_SHMGET_SCOREBOARD`. Defining one of those two (as well

as their companions `HAVE_MMAP` and `HAVE_SHMGET` respectively) enables the supplied shared memory code. If your system has another type of shared memory, edit the file `src/main/http_main.c` and add the hooks necessary to use it in Apache. (Send us back a patch too please.)

Historical note: The Linux port of Apache didn't start to use shared memory until version 1.2 of Apache. This oversight resulted in really poor and unreliable behaviour of earlier versions of Apache on Linux.

## **DYNAMIC\_MODULE\_LIMIT**

If you have no intention of using dynamically loaded modules (you probably don't if you're reading this and tuning your server for every last ounce of performance) then you should add - `DDYNAMIC_MODULE_LIMIT=0` when building your server. This will save RAM that's allocated only for supporting dynamically loaded modules.



Here is a system call trace of Apache 2.0.38 with the worker MPM on Solaris 8. This trace was collected using:

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

The `-l` option tells `truss` to log the ID of the LWP (lightweight process--Solaris's form of kernel-level thread) that invokes each system call.

Other systems may have different system call tracing utilities such as `strace`, `kttrace`, or `par`. They all produce similar output.

In this trace, a client has requested a 10KB static file from the `httpd`. Traces of non-static requests or requests with content negotiation look wildly different (and quite ugly in some cases).

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

In this trace, the listener thread is running within LWP #67.

Note the lack of `accept (2)` serialization. On this particular platform, the worker MPM uses an unserialized `accept` by default unless it is listening on multiple ports.

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

Upon accepting the connection, the listener thread wakes up a worker thread to do the request processing. In this trace, the worker thread that handles the request is mapped to LWP #65.

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

In order to implement virtual hosts, Apache needs to know the local socket address used to accept the connection. It is possible to eliminate this call in many situations (such as when there are no virtual hosts, or when [Listen](#) directives are used which do not have wildcard addresses). But no effort has yet been made to do these optimizations.

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

The `brk(2)` calls allocate memory from the heap. It is rare to see these in a system call trace, because the `httpd` uses custom memory allocators (`apr_pool` and `apr_bucket_alloc`) for most request processing. In this trace, the `httpd` has just been started, so it must call `malloc(3)` to get the blocks of raw memory with which to create the custom memory allocators.

```
/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
```

Next, the worker thread puts the connection to the client (file descriptor 9) in non-blocking mode. The `setsockopt(2)` and `getsockopt(2)` calls are a side-effect of how Solaris's `libc` handles `fcntl(2)` on sockets.

```
/65:    read(9, "GET /10k.html", 8000)    = 97
```

The worker thread reads the request from the client.

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

This httpd has been configured with Options FollowSymLinks and AllowOverride None. Thus it doesn't need to lstat(2) each directory in the path leading up to the requested file, nor check for .htaccess files. It simply calls stat(2) to verify that the file: 1) exists, and 2) is a regular file, not a directory.

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

In this example, the httpd is able to send the HTTP response header and the requested file with a single sendfilev(2) system call. Sendfile semantics vary among operating systems. On some other systems, it is necessary to do a write(2) or writev(2) call to send the headers before calling sendfile(2).

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

This write(2) call records the request in the access log. Note that one thing missing from this trace is a time(2) call. Unlike Apache 1.3, Apache 2.x uses gettimeofday(3) to look up the time. On some operating systems, like Linux or Solaris, gettimeofday has an optimized implementation that doesn't require as much overhead as a typical system call.

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

The worker thread does a lingering close of the connection.

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

Finally the worker thread closes the file that it has just delivered

and blocks until the listener assigns it another connection.

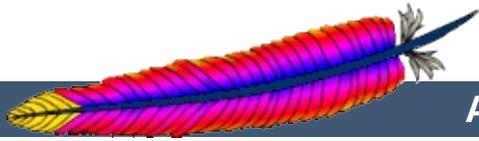
```
/67:    accept(3, 0x001FEB74, 0x001FEB94, 1) (sleeping...)
```

Meanwhile, the listener thread is able to accept another connection as soon as it has dispatched this connection to a worker thread (subject to some flow-control logic in the worker MPM that throttles the listener if all the available workers are busy). Though it isn't apparent from this trace, the next `accept(2)` can (and usually does, under high load conditions) occur in parallel with the worker thread's handling of the just-accepted connection.

---

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](#)

# URL Rewriting Guide

Originally written by  
*Ralf S. Engelschall* <[rse@apache.org](mailto:rse@apache.org)>  
December 1997

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.



---

The Apache module `mod_rewrite` is a killer one, i.e. it is a really sophisticated module which provides a powerful way to do URL manipulations. With it you can do nearly all types of URL manipulations you ever dreamed about. The price you have to pay is to accept complexity, because `mod_rewrite`'s major drawback is that it is not easy to understand and use for the beginner. And even Apache experts sometimes discover new aspects where `mod_rewrite` can help.

In other words: With `mod_rewrite` you either shoot yourself in the foot the first time and never use it again or love it for the rest of your life because of its power. This paper tries to give you a few initial success events to avoid the first case by presenting already invented solutions to you.



## Practical Examples

Here come a lot of practical solutions I've either invented myself or collected from other people's solutions in the past. Feel free to learn the black magic of URL rewriting from these examples.

ATTENTION: Depending on your server-configuration it can 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. It avoid problems.



## Canonical URLs

### Description:

On some webservers 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]
```

## Canonical Hostnames

### 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} !^www\.example\.com [NC]
RewriteCond %{HTTP_HOST} !^$
```

```
RewriteCond %{SERVER_PORT} !^80$
RewriteRule ^/(.*) http://www.example.com:%{SERVER_P

# And for a site running on port 80
RewriteCond %{HTTP_HOST} !^www\.example\.com [NC]
RewriteCond %{HTTP_HOST} !^$
RewriteRule ^/(.*) http://www.example.com/$1 [L,R]
```

## Moved DocumentRoot

### Description:

Usually the [DocumentRoot](#) of the webserver directly relates to the URL "/". But often this data is not really of top-level priority, it is perhaps just one entity of a lot of data pools. For instance at our Intranet sites there are /e/www/ (the homepage for WWW), /e/sww/ (the homepage for the Intranet) etc. Now because the data of the [DocumentRoot](#) stays at /e/www/ we had to make sure that all inlined images and other stuff inside this data pool work for subsequent requests.

### Solution:

We redirect the URL / to /e/www/:

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

Note that this can also be handled using the [RedirectMatch](#) directive:

```
RedirectMatch ^/$ http://example.com/e/www/
```

## Trailing Slash Problem

## Description:

Every webmaster can sing a song about the problem of the trailing slash on URLs referencing directories. If they are missing, the server dumps an error, because if you say `/~quux/foo` instead of `/~quux/foo/` then the server searches for a *file* named `foo`. And because this file is a directory it complains. Actually it tries to fix it itself in most of the cases, but sometimes this mechanism need to be emulated by you. For instance after you have done a lot of complicated URL rewritings to CGI scripts etc.

## 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]
```

The crazy and lazy can even do the following in the top-level `.htaccess` file of their homedir. But notice that this creates some processing overhead.

```
RewriteEngine on
```

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

## Webcluster through Homogeneous URL Layout

### Description:

We want to create a homogeneous and consistent URL layout over all WWW servers on a Intranet webcluster, i.e. all URLs (per definition server local and thus server dependent!) become actually *server independent!* What we want is to give the WWW namespace a consistent server-independent layout: no URL should have to include any physically correct target server. The cluster itself should drive us automatically to the physical target host.

### Solution:

First, the knowledge of the target servers come from (distributed) external maps which contain information where our users, groups and entities stay. 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 the URL is not locally valid to a server. The following ruleset does this for us by 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\
```

## Move Homedirs to Different Webserver

### 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]
```

## Structured Homedirs

### Description:

Some sites with thousands of users usually 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 exactly the above layout.

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

## Filesystem Reorganization

### 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 be later accessible via FTP as well, and I didn't want any Web or CGI stuff to be 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 announced URL `/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 forth 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

## NCSA imagemap to Apache mod\_imap

### Description:

When switching from the NCSA webserver to the more modern Apache webserver a lot of people want a smooth transition. So they want pages which use their old NCSA imagemap program to work under Apache with the modern [mod\\_imap](#). The problem is that there are a lot of hyperlinks around which reference the imagemap program via `/cgi-bin/imagemap/path/to/page.map`. Under Apache this has to read just `/path/to/page.map`.

### Solution:

We use a global rule to remove the prefix on-the-fly for all requests:

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

## Search pages in more than one directory

### 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]

```

## Set Environment Variables According To URL Parts

### 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=([^/]+)/(.*)  $1/$3 [E=STATUS:$2]

```

## Virtual User Hosts

## 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\.com(.*) /home/$1$2
```

## Redirect Homedirs For Foreigners

### 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
```

## Redirect Failing URLs To Other Webserver

### 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 notice 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 error 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 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 a safe way. But it does a performance impact on the webserver, because for every request there is one more internal subrequest. So, if your webserver runs on a powerful CPU, use this one. If it is a slow machine, use the first approach or better a [ErrorDocument](#) CGI-script.

## Extended Redirection

## Description:

Sometimes we need more control (concerning the character escaping mechanism) of URLs on redirects. Usually the Apache kernels URL escape function also escapes anchors, i.e. URLs like "url#anchor". You cannot use this directly on redirects with `mod_rewrite` because the `uri_escape()` function of Apache would also escape the hash character. How can we redirect to such a URL?

## Solution:

We have to use a kludge by the use of a NPH-CGI script which does the redirect itself. Because here no escaping is done (NPH=non-parseable headers). First we introduce a new URL scheme `xredirect`: by the following per-server config-line (should be one of the last rewrite rules):

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

This forces all URLs prefixed with `xredirect`: to be piped through the `nph-xredirect.cgi` program. And this program just looks like:

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

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

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

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

##EOF##
```

This provides you with the functionality to do redirects to all URL schemes, i.e. including the one which are not directly accepted by `mod_rewrite`. For instance you can now also redirect to `news:newsgroup` via

```
RewriteRule ^anyurl  xredirect:news:newsgroup
```

Notice: You have not to put `[R]` or `[R, L]` to the above rule because the `xredirect:` need to be expanded later by our special "pipe through" rule above.

## Archive Access Multiplexer

### Description:

Do you know the great CPAN (Comprehensive Perl Archive Network) under <http://www.perl.com/CPAN>? This does a redirect to one of several FTP servers around the world which carry a CPAN mirror and is approximately near the location of

the requesting client. Actually this can be called an FTP access multiplexing service. While CPAN runs via CGI scripts, how can a similar approach implemented via [mod\\_rewrite](#)?

### Solution:

First we notice that from 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##
```

## Time-Dependent Rewriting

### 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...

## **Backward Compatibility for YYYY to XXXX migration**

### **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]
```

## 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 a maximum version for the latest Netscape variants, a minimum version for the Lynx browsers and a average feature version for all others.

### 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 condig does the following: If the HTTP header "User-Agent" begins with "Mozilla/3", the page `foo.html` is rewritten to `foo.NS.html` and 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 by 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 webpages 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 webserver we could use the program webcopy which acts similar via HTTP. But both techniques have one major drawback: The local copy is always just as up-to-date as often we run the program. It would be much better if the mirror is not a static one we have to establish explicitly. Instead we want a dynamic mirror with data which gets updated automatically when there is need (updated data on the remote host).

### **Solution:**

To provide this feature we map the remote webpage or even the complete remote webarea 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 webserver (`www.quux-corp.dom`), while actually keeping and maintaining its data on a (internal) Intranet webserver (`www2.quux-corp.dom`) which is protected by a firewall. The trick is that on the external webserver we retrieve the requested data on-the-fly from the internal one.

### Solution:

First, we have to make sure that our firewall still protects the internal webserver and that only the external webserver is allowed to retrieve data from it. For a packet-filtering firewall we could for instance 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 a lot of possible solutions for this problem. We will discuss first a commonly known DNS-based variant and then the special one with `mod_rewrite`:

### 1. DNS Round-Robin

The simplest method for load-balancing is to use the DNS round-robin feature of BIND. 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 entry:

```
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.
```

Notice that this seems wrong, but is actually an intended feature of BIND and can be used in this way. However, now when `www.foo.com` gets resolved, BIND gives out `www0-wwww6` - but in a slightly permuted/rotated order

every time. This way the clients are spread over the various servers. But notice that this not a perfect load balancing scheme, because DNS resolve information gets cached by the other nameservers on the net, so once a client has resolved `www.foo.com` to a particular `wwwN.foo.com`, all subsequent requests also go to this particular name `wwwN.foo.com`. But the final result is ok, because the total sum of the requests are really 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 in conjunction with auxiliary tools which provides a real load-balancing for 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 just pushed through the 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 completely done on the other machines. This is the essential point.

#### 4. Hardware/TCP Round-Robin

There is a hardware solution available, too. Cisco has a beast called LocalDirector which does a load balancing at the TCP/IP level. Actually this is some sort of a circuit level gateway in front of a webcluster. If you have enough money and really need a solution with high performance, use this one.

## New MIME-type, New Service

### Description:

On the net there are a lot of nifty CGI programs. But their usage is usually boring, so a lot of webmaster 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\_STRINGS) as their input. First, let us configure a new file type with extension `.scgi` (for secure CGI) which will be processed by the popular `cgiwrap` program. The problem here is that for instance we use a Homogeneous URL Layout (see above) a file inside the user homedirs has the URL `/u/user/foo/bar.scgi`. But `cgiwrap` needs the URL in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uqe]/([^/]+)/\.www/(.+)\.scgi(.*?) ...  
... /internal/cgi/user/cgiwrap/~$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 `wwwidx` (which runs Glimpse on a URL subtree). We have to provide the URL area to these programs so they know on which area they have to act on. But usually this ugly, because they are all the times still requested from that areas, i.e.

typically we would run the `swwidx` 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 the area, 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.

## 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-
```

## **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 webserver if missing. This way you can have CGI-generated pages which are statically served unless one (or a cronjob) 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 to `page.html` leads to a internal run of a corresponding `page.cgi` if `page.html` is still missing or has filesize null. The trick here is that `page.cgi` is a usual CGI

script which (additionally to its STDOUT) writes its output to the file `page.html`. Once it was run, the server sends out the data of `page.html`. When the webmaster wants to force a refresh the contents, he just removes `page.html` (usually done by a cronjob).

## Document With Autorefresh

### Description:

Wouldn't it be nice while creating a complex webpage if the webbrowser would automatically refresh the page every time we write a new version from within our editor? Impossible?

### Solution:

No! We just combine the MIME multipart feature, the webserver NPH feature and the URL manipulation power of [mod\\_rewrite](#). First, we establish a new URL feature: Adding just `:refresh` to any URL causes this to be refreshed every time it gets 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, $x, $mtime) = stat($file);
    return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ($n = 0; $n < $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 dozens virtual hosts. But when you are an ISP and have hundreds of virtual hosts to provide this feature is not the best choice.

### Solution:

To provide this feature we map the remote webpage or even the complete remote webarea to our namespace by the use of 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 puposes
```

```
RewriteRule ^(/.*)$ %1/$1 [E=VHOST:${lowercase:%{HTTP_H
```

```
:
```



## 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$ -
```

## 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 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]
```

## Special Authentication Variant

### Description:

Sometimes a very special authentication is needed, for instance a authentication which checks for a set of explicitly configured users. Only these should receive access and without explicit prompting (which would occur when using the 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).



## 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 $_;
}
```

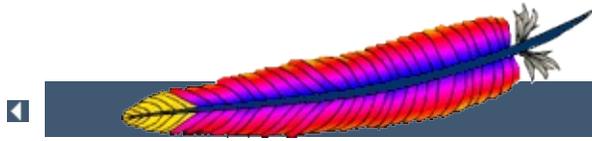
This is a demonstration-only example and just rewrites all URLs `/~quux/foo/...` to `/~quux/bar/...`. Actually you can program whatever you like. But notice that while such

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](#)



| | [FAQ](#) | |

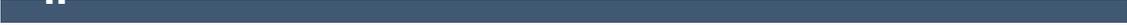


## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

ServerPath





IP

IP

IP DNS

IP

- 

- SSL

- 

SSL

IP



```
core DocumentRoot
      NameVirtualHost
      ServerAlias
      ServerName
      ServerPath
      VirtualHost
      <VirtualHost>
```

```
IP ()
NameVirtualHost * IP
*:80 NameVirtualHost
      IP
      <VirtualHost>
NameVirtualHost (IP
      ServerName
      <VirtualHost>
```

```
ServerName DocumentRoot
```

```
www.domain.tld
```

```
www.other
```

```
httpd.conf
```

```
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 domain.tld *.domain.tld
```

domain.tld www.domain.tld  
ServerName ServerAlias

<VirtualHost>  
(<VirtualHost>)

<NameVirtualHost> IP  
<VirtualHost> ServerName ServerAlias  
IP

IP NameVirtualHost  
<VirtualHost>



?

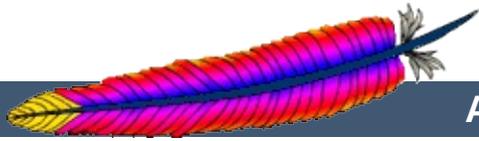
## ServerPath

```
NameVirtualHost 111.22.33.44

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

```
? "/domain" URI
http://www.domain.tld/domain/ Host
http://www.domain.tld/

http://w
"file.html" " ../icons/image.gif") /domain/
(: "http://www.domain.tld/domain/misc/file.html"
"/domain/misc/file.html")
```



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



## Apache HTTP Server Version 2.0

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

# Apache IP-based Virtual Host Support

## See also

[Name-based Virtual Hosts Support](#)



## System Requirements

As the term *IP-based* indicates, the server **must have a different IP address for each IP-based virtual host**. This can be achieved by the machine having several physical network connections, or by use of virtual interfaces which are supported by most modern operating systems (see system documentation for details, these are frequently called "ip aliases", and the "ifconfig" command is most commonly used to set them up).



## How to Setup Apache

There are two ways of configuring apache to support multiple hosts. Either by running a separate `httpd` daemon for each hostname, or by running a single daemon which supports all the virtual hosts.

Use multiple daemons when:

- There are security partitioning issues, such as company1 does not want anyone at company2 to be able to read their data except via the web. In this case you would need two daemons, each running with different `User`, `Group`, `Listen`, and `ServerRoot` settings.
- You can afford the memory and `file descriptor requirements` of listening to every IP alias on the machine. It's only possible to `Listen` to the "wildcard" address, or to specific addresses. So if you have a need to listen to a specific address for whatever reason, then you will need to listen to all specific addresses. (Although one `httpd` could listen to N-1 of the addresses, and another could listen to the remaining address.)

Use a single daemon when:

- Sharing of the `httpd` configuration between virtual hosts is acceptable.
- The machine services a large number of requests, and so the performance loss in running separate daemons may be significant.



## Setting up multiple instances

Create a separate [httpd](#) installation for each virtual host. For each installation, use the [Listen](#) directive in the configuration file to select which IP address (or virtual host) that daemon services. e.g.

```
Listen www.smallco.com:80
```

It is recommended that you use an IP address instead of a hostname (see [DNS caveats](#)).



## Setting up a single instance with virtual hosts

For this case, a single httpd will service requests for the main server and all the virtual hosts. The `VirtualHost` directive in the configuration file is used to set the values of `ServerAdmin`, `ServerName`, `DocumentRoot`, `ErrorLog` and `TransferLog` or `CustomLog` configuration directives to different values for each virtual host. e.g.

```
<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>
```

It is recommended that you use an IP address instead of a hostname (see [DNS caveats](#)).

Almost **any** configuration directive can be put in the `VirtualHost` directive, with the exception of directives that control process creation and a few other directives. To find out if a directive can be used in the `VirtualHost` directive, check the [Context](#) using the [directive index](#).

`SuexecUserGroup` may be used inside a `VirtualHost` directive if the [suEXEC wrapper](#) is used.

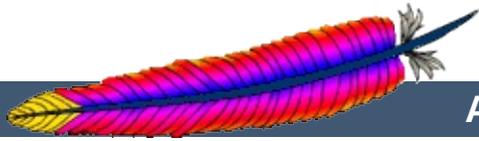
**SECURITY:** When specifying where to write log files, be aware of some security risks which are present if anyone other than the user that starts Apache has write access to the directory where

they are written. See the [security tips](#) document for details.

---

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](#) > [Virtual Hosts](#)

# Dynamically Configured Mass Virtual Hosting

This document describes how to efficiently serve an arbitrary number of virtual hosts with the Apache httpd webserver.



The techniques described here are of interest if your `httpd.conf` contains many `<VirtualHost>` sections that are substantially the same, for example:

```
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>
# blah blah blah
<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>
```

The basic idea is to replace all of the static `<VirtualHost>` configurations with a mechanism that works them out dynamically. This has a number of advantages:

1. Your configuration file is smaller, so Apache starts more quickly and uses less memory.
2. Adding virtual hosts is simply a matter of creating the appropriate directories in the filesystem and entries in the DNS - you don't need to reconfigure or restart Apache.

The main disadvantage is that you cannot have a different log file for each virtual host; however, if you have many virtual hosts, doing this can be a bad idea anyway, because of the number of file descriptors needed. It is better to log to a pipe or a fifo, and arrange for the process at the other end to distribute the logs to the customers. (This can also be used to accumulate statistics,

etc.).



A virtual host is defined by two pieces of information: its IP address, and the contents of the Host : header in the HTTP request. The dynamic mass virtual hosting technique used here is based on automatically inserting this information into the pathname of the file that is used to satisfy the request. This can be most easily done by using [mod\\_vhost\\_alias](#) with Apache 2.0. Alternatively, [mod\\_rewrite](#) can be used. Both of these modules are disabled by default; you must enable one of them when configuring and building Apache if you want to use this technique.

A couple of things need to be `faked' to make the dynamic virtual host look like a normal one. The most important is the server name, which is used by Apache to generate self-referential URLs etc. It is configured with the ServerName directive, and it is available to CGIs via the SERVER\_NAME environment variable. The actual value used at run time is controlled by the [UseCanonicalName](#) setting. With UseCanonicalName Off, the server name is taken from the contents of the Host : header in the request. With UseCanonicalName DNS, it is taken from a reverse DNS lookup of the virtual host's IP address. The former setting is used for name-based dynamic virtual hosting, and the latter is used for IP-based hosting. If Apache cannot work out the server name because there is no Host : header, or the DNS lookup fails, then the value configured with ServerName is used instead.

The other thing to `fake' is the document root (configured with DocumentRoot and available to CGIs via the DOCUMENT\_ROOT environment variable). In a normal configuration, this is used by the core module when mapping URIs to filenames, but when the server is configured to do dynamic virtual hosting, that job must be taken over by another module (either [mod\\_vhost\\_alias](#) or [mod\\_rewrite](#)), which has a different way of doing the mapping.

Neither of these modules is responsible for setting the `DOCUMENT_ROOT` environment variable so if any CGIs or SSI documents make use of it, they will get a misleading value.



## Example 2: Name-based Virtual Hosts

This extract from `httpd.conf` implements the virtual host arrangement outlined in the [Motivation](#) section above, but in a generic fashion using `mod_vhost_alias`.

```
# get the server name from the Host: header
UseCanonicalName Off

# this log format can be split per-virtual-host based on the
# first field
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# include the server name in the filenames used to satisfy
# requests
VirtualDocumentRoot /www/hosts/%0/docs
VirtualScriptAlias /www/hosts/%0/cgi-bin
```

This configuration can be changed into an IP-based virtual hosting solution by just turning `UseCanonicalName Off` into `UseCanonicalName DNS`. The server name that is inserted into the filename is then derived from the IP address of the virtual host.



## VirtualHosts for Homepages System

This is an adjustment of the above system, tailored for an ISP's homepages server. Using a slightly more complicated configuration, we can select substrings of the server name to use in the filename so that, for example, the documents for `www.user.isp.com` are found in `/home/user/`. It uses a single `cgi-bin` directory instead of one per virtual host.

```
# all the preliminary stuff is the same as above, then
# include part of the server name in the filenames
VirtualDocumentRoot /www/hosts/%2/docs

# single cgi-bin directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```

There are examples of more complicated `VirtualDocumentRoot` settings in the [mod\\_vhost\\_alias](#) documentation.



## Server

With more complicated setups, you can use Apache's normal `<VirtualHost>` directives to control the scope of the various virtual hosting configurations. For example, you could have one IP address for general customers' homepages, and another for commercial customers, with the following setup. This can, of course, be combined with conventional `<VirtualHost>` configuration sections.

```
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>
```



The configuration changes suggested to turn [the first example](#) into an IP-based virtual hosting setup result in a rather inefficient setup. A new DNS lookup is required for every request. To avoid this overhead, the filesystem can be arranged to correspond to the IP addresses, instead of to the host names, thereby negating the need for a DNS lookup. Logging will also have to be adjusted to fit this system.

```
# get the server name from the reverse DNS of the IP address
UseCanonicalName DNS

# include the IP address in the logs so they may be split
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

# include the IP address in the filenames
VirtualDocumentRootIP /www/hosts/%0/docs
VirtualScriptAliasIP /www/hosts/%0/cgi-bin
```



This extract from `httpd.conf` does the same thing as [the first example](#). The first half is very similar to the corresponding part above, except for some changes, required for backward compatibility and to make the `mod_rewrite` part work properly; the second half configures `mod_rewrite` to do the actual work.

There are a couple of especially tricky bits: by default, `mod_rewrite` runs before other URI translation modules (`mod_alias` etc.) - so if you wish to use these modules, `mod_rewrite` must be configured to accommodate them. Also, some magic is required to do a per-dynamic-virtual-host equivalent of `ScriptAlias`.

```
# get the server name from the Host: header
UseCanonicalName Off

# splittable logs
LogFormat "%{Host}i %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

<Directory /www/hosts>
    # ExecCGI is needed here because we can't force
    # CGI execution in the way that ScriptAlias does
    Options FollowSymLinks ExecCGI
</Directory>

# now for the hard bit

RewriteEngine On

# a ServerName derived from a Host: header may be any case at
all
RewriteMap lowercase int:tolower

## deal with normal documents first:
# allow Alias /icons/ to work - repeat for other aliases
RewriteCond %{REQUEST_URI} !^/icons/
# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/
# do the magic
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%
{SERVER_NAME}}/docs/$1
```

```
## and now deal with CGIs - we have to force a MIME type
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/cgi-
bin/$1 [T=application/x-httpd-cgi]

# that's it!
```



This does the same thing as [the second example](#).

```
RewriteEngine on

RewriteMap lowercase int:tolower

# allow CGIs to work
RewriteCond %{REQUEST_URI} !^/cgi-bin/

# check the hostname is right so that the RewriteRule works
RewriteCond ${lowercase:%{SERVER_NAME}} ^www\.[a-z-
]+\\.isp\.com$

# concatenate the virtual host name onto the start of the URI
# the [C] means do the next rewrite on the result of this one
RewriteRule ^(.+) ${lowercase:%{SERVER_NAME}}$1 [C]

# now create the real file name
RewriteRule ^www\.[a-z-]+\\.isp\.com/(.*) /home/$1/$2

# define the global CGI directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```



This arrangement uses more advanced `mod_rewrite` features to work out the translation from virtual host to document root, from a separate configuration file. This provides more flexibility, but requires more complicated configuration.

The `vhost.map` file should look something like this:

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

The `httpd.conf` should contain the following:

```
RewriteEngine on

RewriteMap lowercase int:tolower

# define the map file
RewriteMap vhost txt:/www/conf/vhost.map

# deal with aliases as above
RewriteCond %{REQUEST_URI} !^/icons/
RewriteCond %{REQUEST_URI} !^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
# this does the file-based remap
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/docs/$1

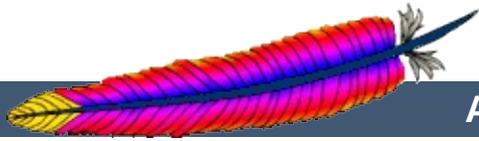
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/cgi-bin/$1 [T=application/x-httpd-cgi]
```

---

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](#) > [Virtual Hosts](#)

## VirtualHost Examples

This document attempts to answer the commonly-asked questions about setting up virtual hosts. These scenarios are those involving multiple web sites running on a single server, via [name-based](#) or [IP-based](#) virtual hosts.



## IP address.

Your server has a single IP address, and multiple aliases (CNAMEs) point to this machine in DNS. You want to run a web server for `www.example1.com` and `www.example2.org` on this machine.

### Note

Creating virtual host configurations on your Apache server does not magically cause DNS entries to be created for those host names. You *must* have the names in DNS, resolving to your IP address, or nobody else will be able to see your web site. You can put entries in your `hosts` file for local testing, but that will work only from the machine with those `hosts` entries.

### Server configuration

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

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

    # Other directives here

</VirtualHost>

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

    # Other directives here

</VirtualHost>
```

The asterisks match all addresses, so the main server serves no

requests. Due to the fact that `www.example1.com` is first in the configuration file, it has the highest priority and can be seen as the *default* or *primary* server. That means that if a request is received that does not match one of the specified `ServerName` directives, it will be served by this first `VirtualHost`.

### Note

You can, if you wish, replace `*` with the actual IP address of the system. In that case, the argument to `VirtualHost` *must* match the argument to `NameVirtualHost`:

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

However, it is additionally useful to use `*` on systems where the IP address is not predictable - for example if you have a dynamic IP address with your ISP, and you are using some variety of dynamic DNS solution. Since `*` matches any IP address, this configuration would work without changes whenever your IP address changes.

The above configuration is what you will want to use in almost all name-based virtual hosting situations. The only thing that this configuration will not work for, in fact, is when you are serving different content based on differing IP addresses or ports.



## Note

Any of the techniques discussed here can be extended to any number of IP addresses.

The server has two IP addresses. On one (172.20.30.40), we will serve the "main" server, server.domain.com and on the other (172.20.30.50), we will serve two or more virtual hosts.

## Server configuration

```
Listen 80

# This is the "main" server running on 172.20.30.40
ServerName server.domain.com
DocumentRoot /www/mainserver

# This is the other address
NameVirtualHost 172.20.30.50

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

    # Other directives here ...
</VirtualHost>

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

    # Other directives here ...
</VirtualHost>
```

Any request to an address other than 172.20.30.50 will be served from the main server. A request to 172.20.30.50 with an unknown hostname, or no Host : header, will be served from www.example1.com.



(such as an internal and external address).

The server machine has two IP addresses (192.168.1.1 and 172.20.30.40). The machine is sitting between an internal (intranet) network and an external (internet) network. Outside of the network, the name `server.example.com` resolves to the external address (172.20.30.40), but inside the network, that same name resolves to the internal address (192.168.1.1).

The server can be made to respond to internal and external requests with the same content, with just one `VirtualHost` section.

### Server configuration

```
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>
```

Now requests from both networks will be served from the same `VirtualHost`.

### Note:

On the internal network, one can just use the name `server` rather than the fully qualified host name `server.example.com`.

Note also that, in the above example, you can replace the list of IP addresses with `*`, which will cause the server to respond the same on all addresses.



You have multiple domains going to the same IP and also want to serve multiple ports. By defining the ports in the "NameVirtualHost" tag, you can allow this to work. If you try using <VirtualHost name:port> without the NameVirtualHost name:port or you try to use the Listen directive, your configuration will not work.

### Server configuration

```
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>
```



## IP-based Virtual Hosting

The server has two IP addresses (172.20.30.40 and 172.20.30.50) which resolve to the names `www.example1.com` and `www.example2.org` respectively.

### Server configuration

```
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>
```

Requests for any address not specified in one of the `<VirtualHost>` directives (such as `localhost`, for example) will go to the main server, if there is one.



The server machine has two IP addresses (172.20.30.40 and 172.20.30.50) which resolve to the names `www.example1.com` and `www.example2.org` respectively. In each case, we want to run hosts on ports 80 and 8080.

### Server configuration

```
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>
```



On some of my addresses, I want to do name-based virtual hosts, and on others, IP-based hosts.

## Server configuration

```
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-based
<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>
```



The following example allows a front-end machine to proxy a virtual host through to a server running on another machine. In the example, a virtual host of the same name is configured on a machine at 192.168.111.2. The [ProxyPreserveHost On](#) directive is used so that the desired hostname is passed through, in case we are proxying multiple hostnames to a single machine.

```
<VirtualHost *:*>
ProxyPreserveHost On
ProxyPass / http://192.168.111.2/
ProxyPassReverse / http://192.168.111.2/
ServerName hostname.example.com
</VirtualHost>
```



## **`_default_` vhosts for all ports**

Catching *every* request to any unspecified IP address and port, *i.e.*, an address/port combination that is not used for any other virtual host.

### **Server configuration**

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

Using such a default vhost with a wildcard port effectively prevents any request going to the main server.

A default vhost never serves a request that was sent to an address/port that is used for name-based vhosts. If the request contained an unknown or no `Host :` header it is always served from the primary name-based vhost (the vhost for that address/port appearing first in the configuration file).

You can use [AliasMatch](#) or [RewriteRule](#) to rewrite any request to a single information page (or script).

## **`_default_` vhosts for different ports**

Same as setup 1, but the server listens on several ports and we want to use a second `_default_` vhost for port 80.

### **Server configuration**

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

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

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

The default vhost for port 80 (which *must* appear before any default vhost with a wildcard port) catches all requests that were sent to an unspecified IP address. The main server is never used to serve a request.

## **`_default_` vhosts for one port**

We want to have a default vhost for port 80, but no other default vhosts.

### **Server configuration**

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

A request to an unspecified address on port 80 is served from the default vhost any other request to an unspecified address and port is served from the main server.



The name-based vhost with the hostname `www.example2.org` (from our [name-based](#) example, setup 2) should get its own IP address. To avoid problems with name servers or proxies who cached the old IP address for the name-based vhost we want to provide both variants during a migration phase. The solution is easy, because we can simply add the new IP address (`172.20.30.50`) to the `VirtualHost` directive.

### Server configuration

```
Listen 80
ServerName www.example1.com
DocumentRoot /www/example1

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40 172.20.30.50>
    DocumentRoot /www/example2
    ServerName www.example2.org
    # ...
</VirtualHost>

<VirtualHost 172.20.30.40>
    DocumentRoot /www/example3
    ServerName www.example3.net
    ServerAlias *.example3.net
    # ...
</VirtualHost>
```

The vhost can now be accessed through the new address (as an IP-based vhost) and through the old address (as a name-based vhost).



We have a server with two name-based vhosts. In order to match the correct virtual host a client must send the correct Host : header. Old HTTP/1.0 clients do not send such a header and Apache has no clue what vhost the client tried to reach (and serves the request from the primary vhost). To provide as much backward compatibility as possible we create a primary vhost which returns a single page containing links with an URL prefix to the name-based virtual hosts.

### Server configuration

```
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>
```

Due to the [ServerPath](#) directive a request to the URL `http://www.sub1.domain.tld/sub1/` is *always* served from the sub1-vhost.

A request to the URL `http://www.sub1.domain.tld/` is only served from the `sub1-vhost` if the client sent a correct `Host :` header. If no `Host :` header is sent the client gets the information page from the primary host.

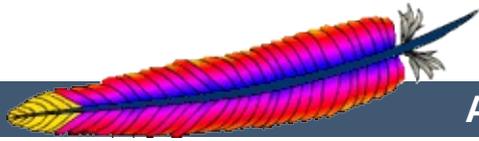
Please note that there is one oddity: A request to `http://www.sub2.domain.tld/sub1/` is also served from the `sub1-vhost` if the client sent no `Host :` header.

The [RewriteRule](#) directives are used to make sure that a client which sent a correct `Host :` header can use both URL variants, *i.e.*, with or without URL prefix.

---

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](#) > [Virtual Hosts](#)

## An In-Depth Discussion of Virtual Host Matching

The virtual host code was completely rewritten in **Apache 1.3**. This document attempts to explain exactly what Apache does when deciding what virtual host to serve a hit from. With the help of the new [NameVirtualHost](#) directive virtual host configuration should be a lot easier and safer than with versions prior to 1.3.

If you just want to *make it work* without understanding how, here are [some examples](#).



There is a *main\_server* which consists of all the definitions appearing outside of `<VirtualHost>` sections. There are virtual servers, called *vhosts*, which are defined by `<VirtualHost>` sections.

The directives `Listen`, `ServerName`, `ServerPath`, and `ServerAlias` can appear anywhere within the definition of a server. However, each appearance overrides the previous appearance (within that server).

The default value of the `Listen` field for *main\_server* is 80. The *main\_server* has no default `ServerPath`, or `ServerAlias`. The default `ServerName` is deduced from the server's IP address.

The *main\_server* `Listen` directive has two functions. One function is to determine the default network port Apache will bind to. The second function is to specify the port number which is used in absolute URIs during redirects.

Unlike the *main\_server*, *vhost* ports *do not* affect what ports Apache listens for connections on.

Each address appearing in the `VirtualHost` directive can have an optional port. If the port is unspecified it defaults to the value of the *main\_server*'s most recent `Listen` statement. The special port `*` indicates a wildcard that matches any port. Collectively the entire set of addresses (including multiple A record results from DNS lookups) are called the *vhost's address set*.

Unless a `NameVirtualHost` directive is used for a specific IP address the first *vhost* with that address is treated as an IP-based *vhost*. The IP address can also be the wildcard `*`.

If name-based *vhosts* should be used a `NameVirtualHost`

directive *must* appear with the IP address set to be used for the name-based vhosts. In other words, you must specify the IP address that holds the hostname aliases (CNAMEs) for your name-based vhosts via a `NameVirtualHost` directive in your configuration file.

Multiple `NameVirtualHost` directives can be used each with a set of `VirtualHost` directives but only one `NameVirtualHost` directive should be used for each specific IP:port pair.

The ordering of `NameVirtualHost` and `VirtualHost` directives is not important which makes the following two examples identical (only the order of the `VirtualHost` directives for *one* address set is important, see below):

```
NameVirtualHost
111.22.33.44
<VirtualHost
111.22.33.44>
# server A
...
</VirtualHost>
<VirtualHost
111.22.33.44>
# server B
...
</VirtualHost>

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

```
<VirtualHost
111.22.33.44>
# server A
</VirtualHost>
<VirtualHost
111.22.33.55>
# server C
...
</VirtualHost>
<VirtualHost
111.22.33.44>
# server B
...
</VirtualHost>
<VirtualHost
111.22.33.55>
# server D
...
</VirtualHost>

NameVirtualHost
111.22.33.44
NameVirtualHost
111.22.33.55
```

(To aid the readability of your configuration you should prefer the left variant.)

After parsing the `VirtualHost` directive, the vhost server is given a default `Listen` equal to the port assigned to the first name in its `VirtualHost` directive.

The complete list of names in the `VirtualHost` directive are treated just like a `ServerAlias` (but are not overridden by any `ServerAlias` statement) if all names resolve to the same address set. Note that subsequent `Listen` statements for this vhost will not affect the ports assigned in the address set.

During initialization a list for each IP address is generated and inserted into an hash table. If the IP address is used in a `NameVirtualHost` directive the list contains all name-based vhosts for the given IP address. If there are no vhosts defined for that address the `NameVirtualHost` directive is ignored and an error is logged. For an IP-based vhost the list in the hash table is empty.

Due to a fast hashing function the overhead of hashing an IP address during a request is minimal and almost not existent. Additionally the table is optimized for IP addresses which vary in the last octet.

For every vhost various default values are set. In particular:

1. If a vhost has no `ServerAdmin`, `Timeout`, `KeepAliveTimeout`, `KeepAlive`, `MaxKeepAliveRequests`, `ReceiveBufferSize`, or `SendBufferSize` directive then the respective value is inherited from the `main_server`. (That is, inherited from whatever the final setting of that value is in the `main_server`.)

2. The "lookup defaults" that define the default directory permissions for a vhost are merged with those of the `main_server`. This includes any per-directory configuration information for any module.
3. The per-server configs for each module from the `main_server` are merged into the vhost server.

Essentially, the `main_server` is treated as "defaults" or a "base" on which to build each vhost. But the positioning of these `main_server` definitions in the config file is largely irrelevant -- the entire config of the `main_server` has been parsed when this final merging occurs. So even if a `main_server` definition appears after a vhost definition it might affect the vhost definition.

If the `main_server` has no `ServerName` at this point, then the hostname of the machine that [httpd](#) is running on is used instead. We will call the *main\_server address set* those IP addresses returned by a DNS lookup on the `ServerName` of the `main_server`.

For any undefined `ServerName` fields, a name-based vhost defaults to the address given first in the `VirtualHost` statement defining the vhost.

Any vhost that includes the magic `_default_` wildcard is given the same `ServerName` as the `main_server`.



The server determines which vhost to use for a request as follows:

## Hash table lookup

When the connection is first made by a client, the IP address to which the client connected is looked up in the internal IP hash table.

If the lookup fails (the IP address wasn't found) the request is served from the `_default_` vhost if there is such a vhost for the port to which the client sent the request. If there is no matching `_default_` vhost the request is served from the `main_server`.

If the IP address is not found in the hash table then the match against the port number may also result in an entry corresponding to a `NameVirtualHost *`, which is subsequently handled like other name-based vhosts.

If the lookup succeeded (a corresponding list for the IP address was found) the next step is to decide if we have to deal with an IP-based or a name-based vhost.

## IP-based vhost

If the entry we found has an empty name list then we have found an IP-based vhost, no further actions are performed and the request is served from that vhost.

## Name-based vhost

If the entry corresponds to a name-based vhost the name list contains one or more vhost structures. This list contains the vhosts in the same order as the `VirtualHost` directives appear in the config file.

The first vhost on this list (the first vhost in the config file with the specified IP address) has the highest priority and catches any request to an unknown server name or a request without a Host : header field.

If the client provided a Host : header field the list is searched for a matching vhost and the first hit on a ServerName or ServerAlias is taken and the request is served from that vhost. A Host : header field can contain a port number, but Apache always matches against the real port to which the client sent the request.

If the client submitted a HTTP/1.0 request without Host : header field we don't know to what server the client tried to connect and any existing ServerPath is matched against the URI from the request. The first matching path on the list is used and the request is served from that vhost.

If no matching vhost could be found the request is served from the first vhost with a matching port number that is on the list for the IP to which the client connected (as already mentioned before).

## **Persistent connections**

The IP lookup described above is only done *once* for a particular TCP/IP session while the name lookup is done on *every* request during a KeepAlive/persistent connection. In other words a client may request pages from different name-based vhosts during a single persistent connection.

## **Absolute URI**

If the URI from the request is an absolute URI, and its hostname and port match the main server or one of the configured virtual hosts *and* match the address and port to which the client sent the

request, then the scheme/hostname/port prefix is stripped off and the remaining relative URI is served by the corresponding main server or virtual host. If it does not match, then the URI remains untouched and the request is taken to be a proxy request.

## Observations

- A name-based vhost can never interfere with an IP-based vhost and vice versa. IP-based vhosts can only be reached through an IP address of its own address set and never through any other address. The same applies to name-based vhosts, they can only be reached through an IP address of the corresponding address set which must be defined with a `NameVirtualHost` directive.
- `ServerAlias` and `ServerPath` checks are never performed for an IP-based vhost.
- The order of name-/IP-based, the `_default_` vhost and the `NameVirtualHost` directive within the config file is not important. Only the ordering of name-based vhosts for a specific address set is significant. The one name-based vhost that comes first in the configuration file has the highest priority for its corresponding address set.
- For security reasons the port number given in a `Host :` header field is never used during the matching process. Apache always uses the real port to which the client sent the request.
- If a `ServerPath` directive exists which is a prefix of another `ServerPath` directive that appears later in the configuration file, then the former will always be matched and the latter will never be matched. (That is assuming that no `Host :` header field was available to disambiguate the two.)
- If two IP-based vhosts have an address in common, the vhost appearing first in the config file is always matched. Such a thing might happen inadvertently. The server will give a

warning in the error logfile when it detects this.

- A `_default_` vhost catches a request only if there is no other vhost with a matching IP address *and* a matching port number for the request. The request is only caught if the port number to which the client sent the request matches the port number of your `_default_` vhost which is your standard Listen by default. A wildcard port can be specified (*i.e.*, `_default_:`\*) to catch requests to any available port. This also applies to `NameVirtualHost *` vhosts.
- The `main_server` is only used to serve a request if the IP address and port number to which the client connected is unspecified and does not match any other vhost (including a `_default_` vhost). In other words the `main_server` only catches a request for an unspecified address/port combination (unless there is a `_default_` vhost which matches that port).
- A `_default_` vhost or the `main_server` is *never* matched for a request with an unknown or missing `Host :` header field if the client connected to an address (and port) which is used for name-based vhosts, *e.g.*, in a `NameVirtualHost` directive.
- You should never specify DNS names in `VirtualHost` directives because it will force your server to rely on DNS to boot. Furthermore it poses a security threat if you do not control the DNS for all the domains listed. There's [more information](#) available on this and the next two topics.
- `ServerName` should always be set for each vhost. Otherwise A DNS lookup is required for each vhost.



**tips**

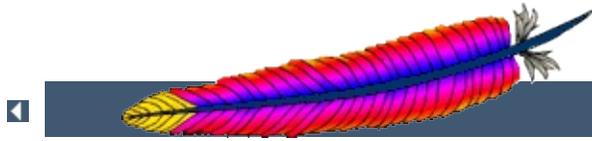
In addition to the tips on the [DNS Issues](#) page, here are some further tips:

- Place all `main_server` definitions before any `VirtualHost` definitions. (This is to aid the readability of the configuration -- the post-config merging process makes it non-obvious that definitions mixed in around virtual hosts might affect all virtual hosts.)
- Group corresponding `NameVirtualHost` and `VirtualHost` definitions in your configuration to ensure better readability.
- Avoid `ServerPaths` which are prefixes of other `ServerPaths`. If you cannot avoid this then you have to ensure that the longer (more specific) prefix vhost appears earlier in the configuration file than the shorter (less specific) prefix (*i.e.*, "`ServerPath /abc`" should appear after "`ServerPath /abc/def`").

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

20 Unix

64

Apache

1. setrlimit()
2. setrlimit(RLIMIT\_NOFILE) (Solaris 2.3)
- 3.
4. stdio 256

:

- <VirtualHost>
- 1 2 Apache

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



---

LogFormat %v :

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost  
CustomLog logs/multiple_vhost_log vhost
```

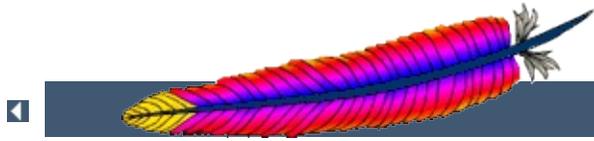
common log format ( ServerName ) (  
[Custom Log Formats](#) )

( 1 ) [split-logfile](#) Apache  
support

:

```
split-logfile < /logs/multiple_vhost_log
```

.log



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

## DNS Apache

This translation may be out of date. Check the English version for recent changes.

Apache

DNS A





()

abc.dom

DNS

Apache 1.2

v

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

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

10.0.0.1 www.abc.dom

10.0.0.2 www.def.dom

def.dom DNS

abc.d

www.def.dom 10.0.0.1

DNS DNS

www.def.dom

10.0.0.1

(http://www.abc.dom/whatever URL

def.dom



---

Apache 1.1  
ServerName  
DNS

Apache httpd IP  
C           gethostname (  
              DNS

DNS  
      OS                                /etc/resolv.cc  
/etc/nsswitch.conf

DNS                               HOSTRESORDER   local  
mod\_env                        CGI man OS



- 
- VirtualHost IP
  - Listen IP
  - ServerName
  - <VirtualHost \_default\_\*>



DNS Apache 1.2

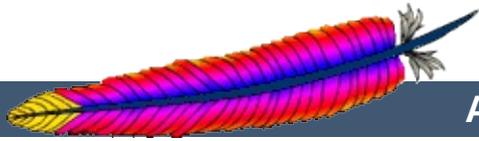
DNS

DNS DNS

DNS )

IP DNS

HTTP/1.1 Host IP



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



## Apache HTTP Server Version 2.0

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

## Support - Frequently Asked Questions



- ["Why can't I ...? Why won't ... work?" What to do in case of problems](#)
- [Whom do I contact for support?](#)

## "Why can't I ...? Why won't ... work?" What to do in case of problems

If you are having trouble with your Apache server software, you should take the following steps:

### Check the errorlog!

Apache tries to be helpful when it encounters a problem. In many cases, it will provide some details by writing one or messages to the server error log. Sometimes this is enough for you to diagnose & fix the problem yourself (such as file permissions or the like). The default location of the error log is `/usr/local/apache2/logs/error_log`, but see the [ErrorLog](#) directive in your config files for the location on your server.

### Check the [FAQ](#)!

The latest version of the Apache Frequently-Asked Questions list can always be found at the main Apache web site.

### Check the Apache bug database

Most problems that get reported to The Apache Group are recorded in the [bug database](#). **Please** check the existing reports, open **and** closed, before adding one. If you find that your issue has already been reported, please *don't* add a "me, too" report. If the original report isn't closed yet, we suggest that you check it periodically. You might also consider contacting the original submitter, because there may be an email exchange going on about the issue that isn't getting recorded in the database.

## Ask in a user support forum

Apache has an active community of users who are willing to share their knowledge. Participating in this community is usually the best and fastest way to get answers to your questions and problems.

[Users mailing list](#)

[#httpd](#) on [Freenode IRC](#) is available for user support issues.

USENET newsgroups:

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

## If all else fails, report the problem in the bug database

If you've gone through those steps above that are appropriate and have obtained no relief, then please *do* let the httpd developers know about the problem by [logging a bug report](#).

If your problem involves the server crashing and generating a core dump, please include a backtrace (if possible). As an example,

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

(Substitute the appropriate locations for your *ServerRoot* and your *httpd* and *core* files. You may have to use *gdb* instead of *dbx*.)

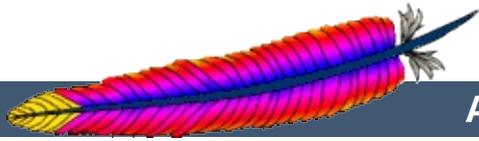
## Whom do I contact for support?

With several million users and fewer than forty volunteer developers, we cannot provide personal support for Apache. For free support, we suggest participating in a [user forum](#).

---

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](#) > [FAQ](#)

## Error Messages - Frequently Asked Questions



- [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**

Apache uses the `sendfile` syscall on platforms where it is available in order to speed sending of responses. Unfortunately, on some systems, Apache will detect the presence of `sendfile` at compile-time, even when it does not work properly. This happens most frequently when using network or other non-standard file-system.

Symptoms of this problem include the above message in the error log and zero-length responses to non-zero-sized files. The problem generally occurs only for static files, since dynamic content usually does not make use of `sendfile`.

To fix this problem, simply use the [EnableSendfile](#) directive to disable `sendfile` for all or part of your server. Also see the [EnableMMAP](#), which can help with similar problems.

## **AcceptEx Failed**

If you get error messages related to the `AcceptEx` syscall on win32, see the [Win32DisableAcceptEx](#) directive.

## **Premature end of script headers**

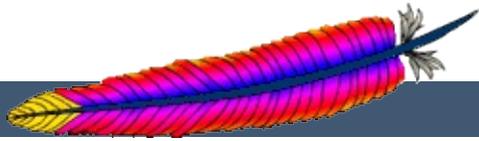
Most problems with CGI scripts result in this message written in the error log together with an `Internal Server Error` delivered to the browser. A guide to helping debug this type of

problem is available in the [CGI tutorial](#).

---

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

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



| [FAQ](#) |



## Apache HTTP 2.0

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

## SSL/TLS :

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

WebHTTPApache SSL

mod\_ssl

The Open Group Research Institute

[Frederick J. Hirsch](#)

[Introducing SSL and Certificates using SSL](#)

[Leay](#) [Web](#)

[Security: A Matter of Trust](#), World Wide Web Journal, Volume 2, Issue 3, Summer 1997

[Frederick Hirsch](#) ()

[Engelschall](#) ([mod\\_ssl](#)) [:

[Apache](#)



---

SSL (:) ([

[AC96\]](#))

0

0



---

(Certificate Authority CA)  
(certificate) (authentication)

1

**1:**



()

( 2 )

**2:**

Common Name ()	CN	SSL URL	CN=www.example.com
Organization or Company ()	O		O=Example Japan K.K.
Organizational Unit ()	OU		OU=Customer Service
City/Locality ()	L		L=Sapporo
State/Province ()	ST		ST=Hokkaido
Country()	C	ISO	C=JP

ASN.1 [X208] [PKCS]  
Encoding Rules (DER) Basic Encoding Rules  
Base64 [ MIME] ASCII  
"Privacy Enhanced Mail" )

### PEM (example.crt)

```
-----BEGIN CERTIFICATE-----
MIIC7jCCAlEgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBqTELMAkGA1UEBhMCWFkx
FTATBgNVBAGTDFNuYWt1IERlc2VydDETMDEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChM0U25ha2UgT21sLCBMdGQxHjAcBgNVBAsTFUN1cnRpbm1jYXR1IEF1dGhv
cm10eTEVMBMGA1UEAxMMU25ha2UgT21sIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTGxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAkGA1UEBhMCWFkxFTATBgNVBAGTDFNuYWt1IERlc2VydDETMDEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChM0U25ha2UgT21sLCBMdGQxHjAcBgNVBAsTD1d1YnN1
cnZlcjBUZWFtMRkwFwYDVQQDEXB3d3cuc25ha2VvaWwuZG9tMR8wHQYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zcH+da+rPTx/DPRp3xGjHZ4GG6pCmvADIEtBtKBFACZ64n+Dy7Np8b
vKR+yy5DGQiijsH1D/j8H1GE+q4TZ80Fk7BNBFazHxFbYI40KMicxdKzdif1yfaa
lWoANFlAz1SdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWGSAGG+EIBAQQEAwIAQDANBgkqhkiG9w0BAQQFAA0B
gQAZUIHAL4D09oE6Lv2k56Gp380BDuILvwLg1v1KL8mQR+KFjghCrtppqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdkPfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHzICxBVC11nHyYGjDuAMhe3961YAn8bCl1d1/L4NMGBCQ==
-----END CERTIFICATE-----
```

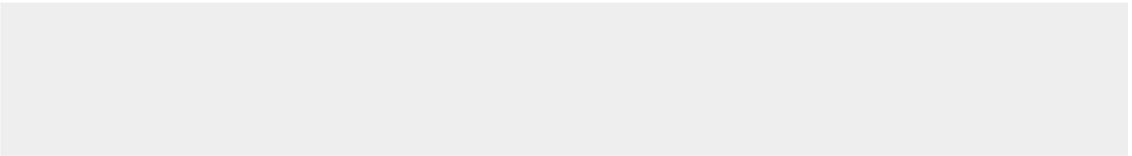
[Thawte](#)

[VeriSign](#)

:

- 
- 
- 

CRL)



## Secure Sockets Layer (TCP/IP) (HTTP) SSL

SSL

### 4: SSL

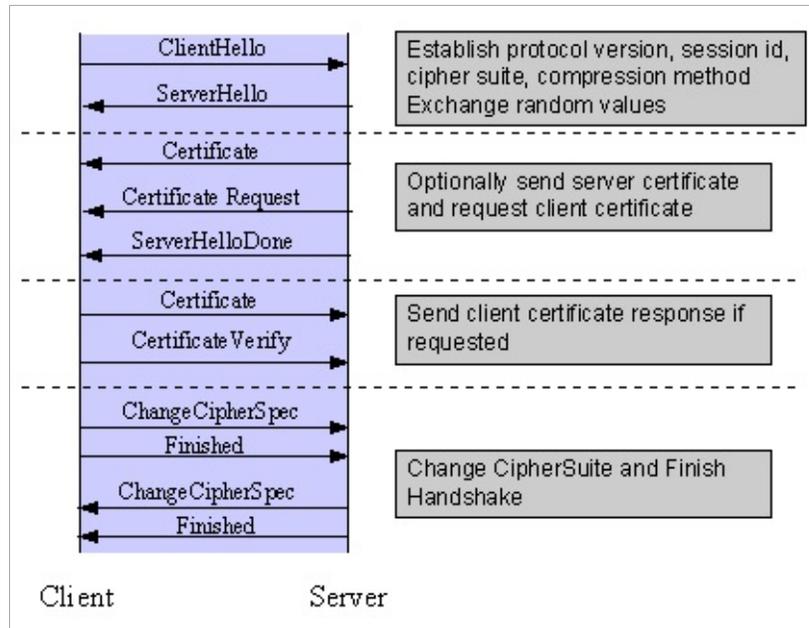
SSL v2.0	Vendor Standard (Netscape Corp. ) [ <a href="#">SSL2</a> ]	SSL	- NS Navigator 1.x/2.x - MS IE 3.x - Lynx/2.8+OpenSSL
SSL v3.0	Expired Internet Draft (Netscape Corp. ) [ <a href="#">SSL3</a> ]		- NS Navigator 2.x/3.x/4.x - MS IE 3.x/4.x - Lynx/2.8+OpenSSL
TLS v1.0	Proposed Internet Standard (IETF ) [ <a href="#">TLS1</a> ]	MAC HMAC block padding 3.0	- Lynx/2.8+OpenSSL

#### 4SSL SSL 3.0

SSL 3.0 Internet Engineering Task Force (IETF)  
Transport Layer Security [[TLS](#)]

1

SSL



## 1: SSL

:

- 1.
- 2.
- 3.
- 4.

- 
- 
- Message Authentication Code (MAC)

Hellman

SSL

()

:

- 
- - 40-bit RC4
  - 128-bit RC4
- CBC
  - 40 bit RC2
  - 40 bit DES
  - 56 bit DES
  - 168 bit Triple-DES
  - Idea (128 bit )
  - Fortezza (96 bit )

CBC (Cipher Block Chaining)  
Encryption Standard) [[AC96](#), ch12]  
Idea

DES40 3DES\_EDE

RC2

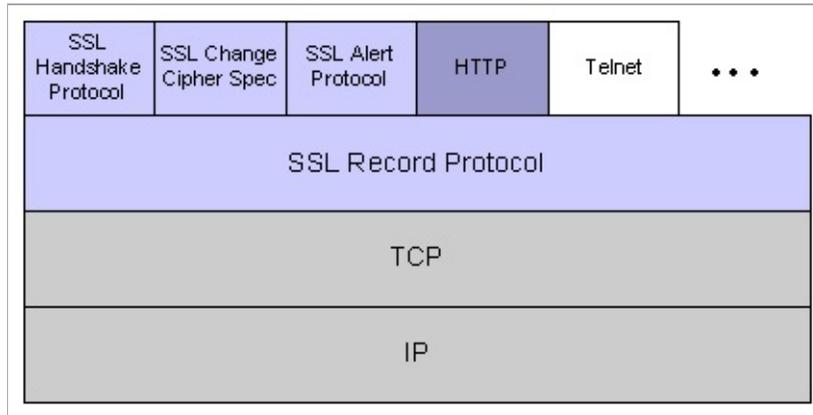
- 
- MD5 (128-bit )
- Secure Hash Algorithm (SHA-1) (160-bit )

Message Authentication Code (MAC)

:

- SSL SSL
- SSL
- SSL SSL

## 2 SSL

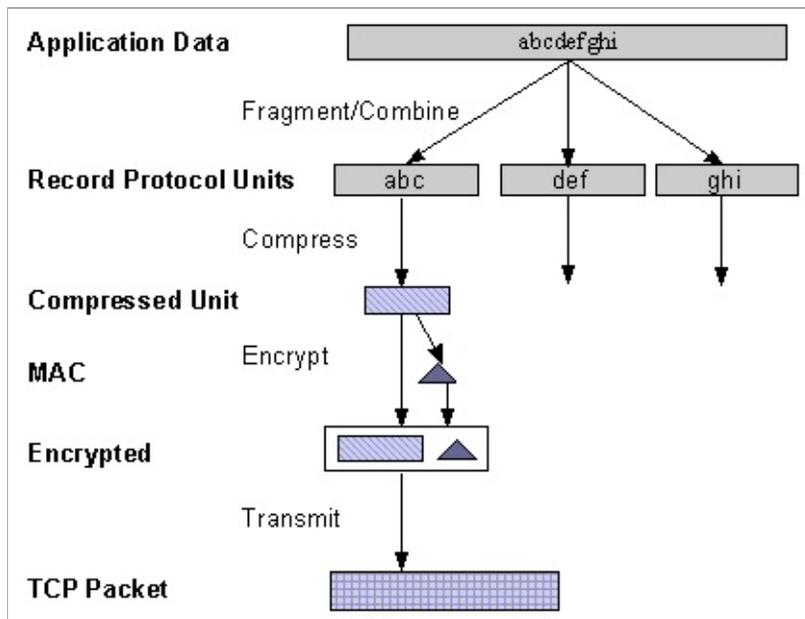


## 2: SSL

SSL

## 3 SSL

S



## 3 : SSL

## HTTP

SSL HTTP

HTTPS URL

mod\_ssl Apache

http https (443) HT



---

**[AC96]**

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

**[X208]**

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

**[X509]**

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

**[PKCS]**

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

**[MIME]**

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

**[SSL2]**

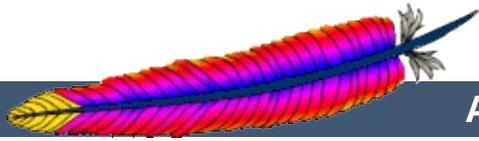
Kipp E.B. Hickman, *The SSL Protocol*, 1995. See [http://www.netscape.com/eng/security/SSL\\_2.html](http://www.netscape.com/eng/security/SSL_2.html).

**[SSL3]**

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

## [TLS1]

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



[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
<b>Apache-SSL 1.x &amp; <code>mod_ssl</code> 2.0.x compatibility:</b>	
<code>SSLEnable</code>	<code>SSL Engine on</code>
<code>SSLDisable</code>	<code>SSL Engine off</code>
<code>SSLLogFile <i>file</i></code>	<code>SSL Log <i>file</i></code>
<code>SSLRequiredCiphers <i>spec</i></code>	<code>SSL Cipher Suite <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>SSL Options +FakeBasicAuth</code>
<code>SSLCacheServerPath <i>dir</i></code>	-
<code>SSLCacheServerPort <i>integer</i></code>	-
<b>Apache-SSL 1.x compatibility:</b>	
<code>SSLExportClientCertificates</code>	<code>SSL Options +ExportCertData</code>

SSLCacheServerRunDir <i>dir</i>	-
<b>Sioux 1.x compatibility:</b>	
SSL_CertFile <i>file</i>	SSLCertificateFile <i>file</i>
SSL_KeyFile <i>file</i>	SSLCertificateKeyFile <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>	SSL Engine <i>flag</i>
SSL_ClientAuth <i>arg</i>	SSLVerifyClient <i>arg</i>
SSL_X509VerifyDepth <i>arg</i>	SSLVerifyDepth <i>arg</i>
SSL_FetchKeyPhraseFrom <i>arg</i>	-
SSL_SessionDir <i>dir</i>	-
SSL_Require <i>expr</i>	-
SSL_CertFileType <i>arg</i>	-
SSL_KeyFileType <i>arg</i>	-
SSL_X509VerifyPolicy <i>arg</i>	-
SSL_LogX509Attributes <i>arg</i>	-
<b>Stronghold 2.x compatibility:</b>	
StrongholdAccelerator <i>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_CRYPT0	-
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](#).

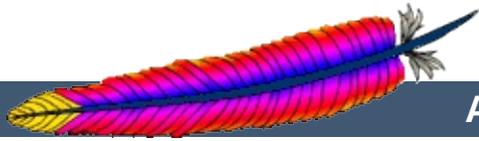
**Table 3: Custom Log Cryptography Function**

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

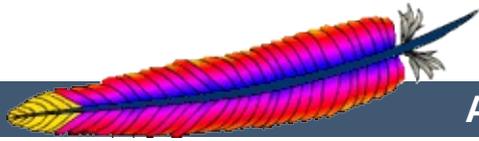
#   Network Access Control
Order               deny,allow
Deny                from all
Allow               192.168.1.0/24

#   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](mailto:comp.infosystems.www.servers.unix) or the mod\_ssl Support Mailing List [modssl-users@modssl.org](mailto:modssl-users@modssl.org). They are collected at this place to avoid answering the same questions over and over.

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



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

## What is the history of mod\_ssl?

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

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

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

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

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

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

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

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



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

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

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

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

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

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

SSLRandomSeed directive.



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

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

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

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

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

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

While you usually just use

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
RewriteEngine on
RewriteRule ^/(.*):SSL$ https://%{SERVER_NAME}/$1 [R,L]
RewriteRule ^/(.*):NOSSL$ http://%{SERVER_NAME}/$1 [R,L]
```

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



- [What are RSA Private Keys, CSRs and Certificates?](#)
- [Is there a difference on startup between a non-SSL-aware Apache and an SSL-aware Apache?](#)
- [How do I create a self-signed SSL Certificate for testing purposes?](#)
- [How do I create a real SSL Certificate?](#)
- [How do I create and use my own Certificate Authority \(CA\)?](#)
- [How can I change the pass-phrase on my private key file?](#)
- [How can I get rid of the pass-phrase dialog at Apache startup time?](#)
- [How do I verify that a private key matches its Certificate?](#)
- [Why do connections fail with an "alert bad certificate" error?](#)
- [Why does my 2048-bit private key not work?](#)
- [Why is client authentication broken after upgrading from SSLeay version 0.8 to 0.9?](#)
- [How can I convert a certificate from PEM to DER format?](#)
- [Why can't I find the getca or getverisign programs mentioned by Verisign, for installing my Verisign certificate?](#)
- [Can I use the Server Gated Cryptography \(SGC\) facility \(aka Verisign Global ID\) with mod\\_ssl?](#)
- [Why do browsers complain that they cannot verify my 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](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](mailto:modssl-users@modssl.org)

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

### 2. *Send a Problem Report to the Apache httpd Users Support Mailing List*

[users@httpd.apache.org](mailto:users@httpd.apache.org)

This is the second way of submitting your problem report. Again, you must subscribe to the list first, but you can then easily discuss your problem with the whole Apache httpd user community.

### 3. *Write a Problem Report in the Bug Database*

[http://httpd.apache.org/bug\\_report.html](http://httpd.apache.org/bug_report.html)

This is the last way of submitting your problem report. You should only do this if you've already posted to the mailing lists, and had no success. Please follow the instructions on the above page *carefully*.

## **What information should I provide when writing a bug report?**

You should always provide at least the following information:

### **Apache and OpenSSL version information**

The Apache version can be determined by running `httpd -v`. The OpenSSL version can be determined by running `openssl version`. Alternatively, if you have Lynx installed, you can run the command `lynx -mime_header http://localhost/ | grep Server` to gather this information in a single step.

## **The details on how you built and installed Apache+mod\_ssl+OpenSSL**

For this you can provide a logfile of your terminal session which shows the configuration and install steps. If this is not possible, you should at least provide the [configure](#) command line you used.

## **In case of core dumps please include a Backtrace**

If your Apache+mod\_ssl+OpenSSL dumps its core, please attach a stack-frame ``backtrace" (see [below](#) for information on how to get this). This information is required in order to find a reason for your core dump.

## **A detailed description of your problem**

Don't laugh, we really mean it! Many problem reports don't include a description of what the actual problem is. Without this, it's very difficult for anyone to help you. So, it's in your own interest (you want the problem be solved, don't you?) to include as much detail as possible, please. Of course, you should still include all the essentials above too.

## **I had a core dump, can you help me?**

In general no, at least not unless you provide more details about the code location where Apache dumped core. What is usually always required in order to help you is a backtrace (see next question). Without this information it is mostly impossible to find the problem and help you in fixing it.

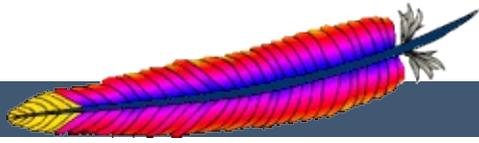
## **How do I get a backtrace, to help find the reason for my core dump?**

Following are the steps you will need to complete, to get a backtrace:

1. Make sure you have debugging symbols available, at least in

Apache. On platforms where you use GCC/GDB, you will have to build Apache+mod\_ssl with ```OPTIM="-g -ggdb3""` to get this. On other platforms at least ```OPTIM="-g""` is needed.

2. Start the server and try to reproduce the core-dump. For this you may want to use a directive like ```CoreDumpDirectory /tmp"` to make sure that the core-dump file can be written. This should result in a `/tmp/core` or `/tmp/httpd.core` file. If you don't get one of these, try running your server under a non-root UID. Many modern kernels do not allow a process to dump core after it has done a `setuid( )` (unless it does an `exec( )`) for security reasons (there can be privileged information left over in memory). If necessary, you can run `/path/to/httpd -X` manually to force Apache to not fork.
3. Analyze the core-dump. For this, run `gdb /path/to/httpd /tmp/httpd.core` or a similar command. In GDB, all you have to do then is to enter `bt`, and voila, you get the backtrace. For other debuggers consult your local debugger manual.



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) > [How-To /](#)

--

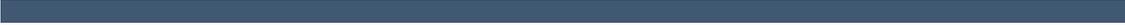
This translation may be out of date. Check the English version for recent changes.



---

<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>





```
( <Dir
```

```
.htaccess
```

```
AllowOverride
```

```
AllowOverride AuthConfig
```



---

`/usr/local/apache/passwd`

Apache [htpasswd](#) Apache

```
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`

`/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

Basic

mod\_auth\_digest

Digest

AuthName

*Realm (:)*

Realm

"Restricted Files"  
Realm

"Res

AuthUserFile

htpasswd

mod\_auth\_dbm

AuthDBMUserFile

Apache

Require



```
( rbowen )
```

```
GroupName: rbowen dpitts sungo rshersey
```

```
htpasswd /usr/local/apache/passwd/passwords dpitts
```

## .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
```

[AuthUserFil](#)



---

## 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
```

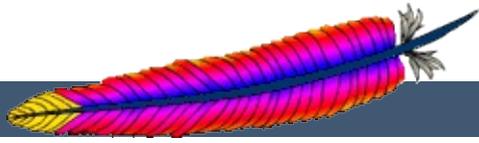
Order        Deny   Allow

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

Allow







| [FAQ](#) |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) > [How-To /](#)

## Apache Tutorial: CGI

This translation may be out of date. Check the English version for recent changes.



---

<u>mod alias</u>	<u>AddHandler</u>
<u>mod cgi</u>	<u>Options</u>
	<u>ScriptAlias</u>

CGI (Common Gateway Interface)



CGI CGI

Apache

## ScriptAlias

ScriptAlias

CGI Apache

ScriptAlias :

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

Apache

ht

Alias

Alias ScriptAlias

Alias ScriptAlias

ScriptAlias

/cgi-bin/

CGI Apache

URL http://dev.rcbowen.com/cgi-bin/test.pl

Apache /usr/local/apache2/cgi-bin/test.pl

Ap

## ScriptAlias CGI

CGI

ScriptAlias

UserDir

CGI

Ad

Options

ExecCGI

## CGI Options

Options

C

```
<Directory /usr/local/apache2/htdocs/somedir>
Options +ExecCGI
```

```
</Directory>
```

CGI  
pl CGI

Apache CGI

:

```
AddHandler cgi-script .cgi .pl
```

## **.htaccess files**

[.htaccess](#) httpd.conf CGI

## **User**

.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

```
Content-type: text/html
```

HTML

CGI

**CGI**

1 CGI

f

```
#!/usr/bin/perl  
print "Content-type: text/html\n\n";  
print "Hello, World.";
```

Perl 1

2 content-type

World."

```
http://www.example.com/cgi-bin/first.pl
```

Hello, World. 1



CGI :

**CGI**  
!

**CGI "POST Method Not Allowed"**

CGI Apache

**"Forbidden"**

[Apache](#)

**"Internal Server Error"**

[Apache](#) "Premature end of script headers"

CGI

HTTP

:

```
chmod a+x first.pl
```

CGI

CGI

( perl) CGI 1 :

```
#!/usr/bin/perl
```

CGI

CGI

```
cd /usr/local/apache2/cgi-bin
./first.pl
```

(perl

Apache

Content-Type HTTP  
end of script headers

CGI

## Suexec

suexec

script headers

suexec apachectl -V SUEXEC\_BIN  
suexec suexec

Ap

suexec

suexec

suexec  
suexec -V suexec



---

CGI

CGI  
WebSite) CGI

CGI

Perl CGI  
Apache

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
foreach $key (keys %ENV) {
    print "$key --> $ENV{$key}<br>";
}
```

## STDIN STDOUT

(STDIN) ( STDOUT)

CGI POST

STDI

(=) (&)

```
name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Squirrel%20Mor
```

URL

GET POST FORM METHOD

CGI



---

CGI

Perl CGI

[CPAN](#)

C CGI



---

CGI CGI

HTML Writers Guide

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

Usenet

-servers

CGI

CGI

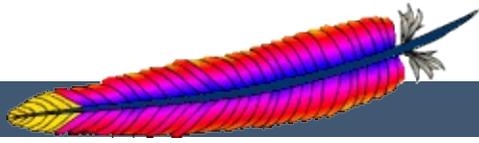
Apache

CGI A

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) > [How-To /](#)

# Apache : Server Side Includes

HTML



---

<u>mod_include</u>	<u>Options</u>
<u>mod_cgi</u>	<u>XBitHack</u>
<u>mod_expires</u>	<u>AddType</u>
	<u>SetOutputFilter</u>
	<u>BrowserMatchNoCase</u>

SSI Server Side Includes

SSI

SSI

SSI SSI



---

SSI (Server Side Includes) HTML  
HTML

SSI



SSI httpd.conf .htaccess :

```
Options +Includes
```

SSI Apache  
Options

SSI Apache

```
AddType text/html .shtml  
AddOutputFilter INCLUDES .shtml
```

SSI

XBitHack :

```
XBitHack on
```

XBitHack SSI Apache  
chmod

```
chmod +x pagename.html
```

XBitHack SSI .html SSI Ap

Windows

Apache SSI HTTP

1. XBitHack Full
2. [mod\\_expires](#)



SSI :

```
<!--#element attribute=value attribute=value ... -->
```

HTML SSI

element

```
<!--#echo var="DATE_LOCAL" -->
```

echo

CGI

config t

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

```
This document last modified <!--#flastmod file="index.html" -->
```

timefmt

CGI

``"

CGI SSI

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



---

## SSI HTML

?

SSI

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

ssi.shtml

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

timefmt

strft

file

/

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

LAST\_MODIFIED



---

config

config

SSI

```
[an error occurred while processing this directive]
```

config errmsg:

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

SSI

config sizefmt  
abbrev





## Apache SSI

Apache 1.2

Apache 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}" -->
```

endif

:

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

*test\_condition*

""

:

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

Macintosh

SSI :

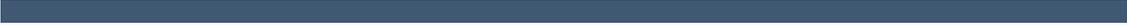
```
<!--#if expr="${Mac} && ${InternetExplorer}" -->
Apologetic text goes here
<!--#else -->
Cool JavaScript code goes here
<!--#endif -->
```

Mac IE

JavaScript Mac

( )



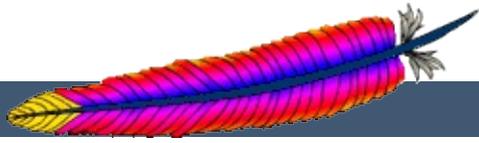


SSI CGI

---

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

| | [FAQ](#) | |



| [FAQ](#) |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) > [How-To /](#)

## Apache : .htaccess

This translation may be out of date. Check the English version for recent changes.

.htaccess





.htaccess ()

```
:  
: .htaccess AccessFileName  
:
```

```
AccessFileName .config
```

.htaccess

AllowOverride

AddDefaultCharset

.htaccess

FileInfo .htaccess

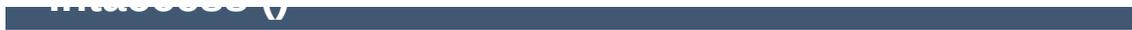
FileInfo

```
:  
: .htaccess  
: FileInfo
```

.htaccess

".htaccess"





.htaccess

.htaccess  
ISP

root

.htaccess

.htaccess

.htaccess

AllowOverride

.htaccess Apache

.htaccess

.htaccess

Apache

/www/htdocs/example Apache

```
/.htaccess
/www/.htaccess
/www/htdocs/.htaccess
/www/htdocs/example/.htaccess
```

.htaccess /www/htdocs/example  
<Directory /www/htdocs/example> :

/www/htdocs/example .htaccess:

```
/www/htdocs/example .htaccess  
AddType text/example .exm
```



## httpd.conf file

```
<Directory /www/htdocs/example>  
  AddType text/example .exm  
</Directory>
```

AllowOverride

none .htaccess

```
AllowOverride None
```



```
.htaccess .htaccess
          .htaccess .h
```

```
:
```

```
/www/htdocs/example1 .htaccess :
```

```
Options +ExecCGI
```

```
(: .htaccess " Options"  
)
```

```
/www/htdocs/example1/example2 .htaccess:
```

```
Options Includes
```

```
.htaccess /www/htdocs/example1/examp:  
CGI Options Includes
```



---

.htaccess

.htaccess

"AllowOverride AuthConfig"

.htaccess :

```
AuthType Basic
AuthName "Password Required"
AuthUserFile /www/passwords/password.file
AuthGroupFile /www/passwords/group.file
Require Group admins
```

AllowOverride AuthConfig



.htaccess

SSI

```
Options +Includes  
AddType text/html shtml  
AddHandler server-parsed shtml
```

AllowOverride Options

SSI [SSI](#)



CGI

:

```
Options +ExecCGI  
AddHandler cgi-script cgi pl
```

CGI

:

```
Options +ExecCGI  
SetHandler cgi-script
```

AllowOverride Options

CGI [CGI](#)



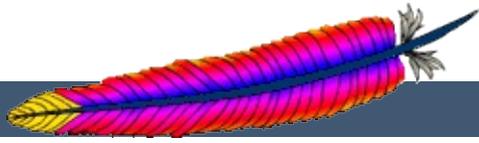
---

.htaccess

AllowOverride

.htaccess

Apache



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) > [How-To /](#)

UserDir  
"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

dissabled

UserDir

UserDir disabled  
UserDir enabled rbowen krietz

UserDir



cgi-bin

<Directory>

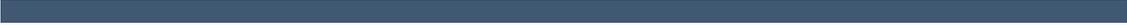
```
<Directory /home/*/public_html/cgi-bin/>  
Options ExecCGI  
SetHandler cgi-script  
</Directory>
```

UserDir public\_html

CGI

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



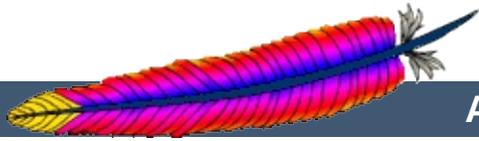


.htacc

---

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

| | [FAQ](#) | |



[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

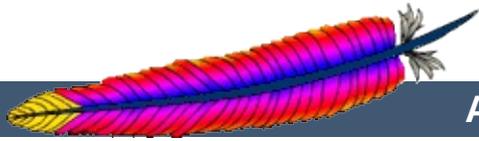
- [Content Negotiation Explained](#) (Apacheweek)
- [Using Apache Imagemaps](#) (Apacheweek)
- [Keeping Your Images from Adorning Other Sites](#) (ApacheToday)
- [Language Negotiation Notes](#) (Alan J. Flavell)

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](#)



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



## Apache HTTP Server Version 2.0

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

# Using Apache with Microsoft Windows

This document explains how to install, configure and run Apache 2.0 under Microsoft Windows. If you find any bugs, or wish to contribute in other ways, please use our [bug reporting page](#).

This document assumes that you are installing a binary distribution of Apache. If you want to compile Apache yourself (possibly to help with development or tracking down bugs), see [Compiling Apache for Microsoft Windows](#).

**Because of the current versioning policies on Microsoft Windows operating system families, this document assumes the following:**

- **Windows NT:** This means all versions of Windows that are based on the Windows NT kernel. Includes Windows NT, Windows 2000, Windows XP and Windows .Net Server 2003.
- **Windows 9x:** This means older, consumer-oriented versions of Windows. Includes Windows 95 (also OSR2), Windows 98 and Windows ME.



## Operating System Requirements

The primary Windows platform for running Apache 2.0 is Windows NT. The binary installer only works with the x86 family of processors, such as Intel and AMD processors. Running Apache on Windows 9x is not thoroughly tested, and it is never recommended on production systems.

On all operating systems, TCP/IP networking must be installed and working. If running on Windows 95, the Winsock 2 upgrade must be installed. Winsock 2 for Windows 95 can be downloaded from [here](#).

On Windows NT 4.0, installing Service Pack 6 is strongly recommended, as Service Pack 4 created known issues with TCP/IP and Winsock integrity that were resolved in later Service Packs.



## Installing Apache for Windows

Information on the latest versions of Apache can be found on the web site of the Apache web server at <http://httpd.apache.org/download.cgi>. There you will find the current release, as well as more recent alpha or beta test versions, and a list of HTTP and FTP mirrors from which you can download the Apache web server. Please use a mirror near to you for a fast and reliable download.

For Windows installations you should download the version of Apache for Windows with the `.msi` extension. This is a single Microsoft Installer file, which contains a ready-to-run version of Apache. There is a separate `.zip` file, which contains only the source code. You can compile Apache yourself with the Microsoft Visual C++ (Visual Studio) tools.



## Installing Apache for Windows

You need Microsoft Installer 1.2 or above for the installation to work. On Windows 9x you can update your Microsoft Installer to version 2.0 [here](#) and on Windows NT 4.0 and 2000 the version 2.0 update can be found [here](#). Windows XP does not need this update.

Note that you cannot install two versions of Apache 2.0 on the same computer with the binary installer. You can, however, install a version of the 1.3 series **and** a version of the 2.0 series on the same computer without problems. If you need to have two different 2.0 versions on the same computer, you have to [compile and install Apache from the source](#).

Run the Apache .msi file you downloaded above. The installation will ask you for these things:

1. **Network Domain.** Enter the DNS domain in which your server is or will be registered in. For example, if your server's full DNS name is server.mydomain.net, you would type mydomain.net here.
2. **Server Name.** Your server's full DNS name. From the example above, you would type server.mydomain.net here.
3. **Administrator's Email Address.** Enter the server administrator's or webmaster's email address here. This address will be displayed along with error messages to the client by default.
4. **For whom to install Apache** Select for All Users, on Port 80, as a Service - Recommended if you'd like your new Apache to listen at port 80 for incoming traffic. It will run as a service (that is, Apache will run even if no one is

logged in on the server at the moment) Select `only` for the `Current User`, on `Port 8080`, when started `Manually` if you'd like to install Apache for your personal experimenting or if you already have another WWW server running on port 80.

5. **The installation type.** Select `Typical` for everything except the source code and libraries for module development. With `Custom` you can specify what to install. A full install will require about 13 megabytes of free disk space. This does *not* include the size of your web site(s).
6. **Where to install.** The default path is `C:\Program Files\Apache Group` under which a directory called `Apache2` will be created by default.

During the installation, Apache will configure the files in the `conf` subdirectory to reflect the chosen installation directory. However, if any of the configuration files in this directory already exist, they will not be overwritten. Instead, the new copy of the corresponding file will be left with the extension `.default`. So, for example, if `conf\httpd.conf` already exists, it will be renamed as `conf\httpd.conf.default`. After the installation you should manually check to see what new settings are in the `.default` file, and if necessary, update your existing configuration file.

Also, if you already have a file called `htdocs\index.html`, it will not be overwritten (and no `index.html.default` will be installed either). This means it should be safe to install Apache over an existing installation, although you would have to stop the existing running server before doing the installation, and then start the new one after the installation is finished.

After installing Apache, you must edit the configuration files in the

conf subdirectory as required. These files will be configured during the installation so that Apache is ready to be run from the directory it was installed into, with the documents server from the subdirectory ht docs. There are lots of other options which you should set before you really start using Apache. However, to get started quickly, the files should work as installed.



## Configuring Apache for Windows

Apache is configured by the files in the `conf` subdirectory. These are the same files used to configure the Unix version, but there are a few different directives for Apache on Windows. See the [directive index](#) for all the available directives.

The main differences in Apache for Windows are:

- Because Apache for Windows is multithreaded, it does not use a separate process for each request, as Apache does on Unix. Instead there are usually only two Apache processes running: a parent process, and a child which handles the requests. Within the child process each request is handled by a separate thread.

The process management directives are also different:

**`MaxRequestsPerChild`**: Like the Unix directive, this controls how many requests a single child process will serve before exiting. However, unlike on Unix, a single process serves all the requests at once, not just one. If this is set, it is recommended that a very high number is used. The recommended default, `MaxRequestsPerChild 0`, causes the child process to never exit.

**Warning: The server configuration file is reread when a new child process is started. If you have modified `httpd.conf`, the new child may not start or you may receive unexpected results.**

**`ThreadsPerChild`**: This directive is new. It tells the server how many threads it should use. This is the maximum number of connections the server can handle at once, so be sure to set this number high enough for your site if you get a lot of

hits. The recommended default is `ThreadsPerChild 50`.

- The directives that accept filenames as arguments must use Windows filenames instead of Unix ones. However, because Apache uses Unix-style names internally, you must use forward slashes, not backslashes. Drive letters can be used; if omitted, the drive with the Apache executable will be assumed.
- While filenames are generally case-insensitive on Windows, URLs are still treated internally as case-sensitive before they are mapped to the filesystem. For example, the `<Location>`, `Alias`, and `ProxyPass` directives all use case-sensitive arguments. For this reason, it is particularly important to use the `<Directory>` directive when attempting to limit access to content in the filesystem, since this directive applies to any content in a directory, regardless of how it is accessed. If you wish to assure that only lowercase is used in URLs, you can use something like:

```
RewriteEngine On
RewriteMap lowercase int:tolower
RewriteCond %{REQUEST_URI} [A-Z]
RewriteRule (.*) ${lowercase:$1} [R,L]
```

- Apache for Windows contains the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the `\Apache2\modules` directory. To activate these or other modules, the new `LoadModule` directive must be used. For example, to activate the status module, use the following (in addition to the status-activating directives in `access.conf`):

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

Information on [creating loadable modules](#) is also available.

- Apache can also load ISAPI (Internet Server Application Programming Interface) extensions (i.e. internet server applications), such as those used by Microsoft IIS and other Windows servers. [More information is available](#). Note that Apache **cannot** load ISAPI Filters.
- When running CGI scripts, the method Apache uses to find the interpreter for the script is configurable using the [ScriptInterpreterSource](#) directive.
- Since it is often difficult to manage files with names like .htaccess in Windows, you may find it useful to change the name of this per-directory configuration file using the [AccessFilename](#) directive.
- Any errors during Apache startup are logged into the Windows event log when running on Windows NT. This mechanism acts as a backup for those situations where Apache cannot even access the normally used error .log file. You can view the Windows event log by using the Event Viewer application on Windows NT 4.0, and the Event Viewer MMC snap-in on newer versions of Windows.

**Note that there is no startup error logging on Windows 9x because no Windows event log exists on those operating systems.**



## Running Apache as a Service

Apache can be run as a service on Windows NT. There is some highly experimental support for similar behavior on Windows 9x.

You can install Apache as a service automatically during the installation. If you chose to install for all users, the installation will create an Apache service for you. If you specify to install for yourself only, you can manually register Apache as a service after the installation. You have to be a member of the Administrators group for the service installation to succeed.

Apache comes with a utility called the Apache Service Monitor. With it you can see and manage the state of all installed Apache services on any machine on your network. To be able to manage an Apache service with the monitor, you have to first install the service (either automatically via the installation or manually).

You can install Apache as a Windows NT service as follows from the command prompt at the Apache `bin` subdirectory:

```
httpd -k install
```

If you need to specify the name of the service you want to install, use the following command. You have to do this if you have several different service installations of Apache on your computer.

```
httpd -k install -n "MyServiceName"
```

If you need to have specifically named configuration files for different services, you must use this:

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

If you use the first command without any special parameters except `-k install`, the service will be called Apache2 and the

configuration will be assumed to be `conf\httpd.conf`.

Removing an Apache service is easy. Just use:

```
httpd -k uninstall
```

The specific Apache service to be uninstalled can be specified by using:

```
httpd -k uninstall -n "MyServiceName"
```

Normal starting, restarting and shutting down of an Apache service is usually done via the Apache Service Monitor, by using commands like `NET START Apache2` and `NET STOP Apache2` or via normal Windows service management. Before starting Apache as a service by any means, you should test the service's configuration file by using:

```
httpd -n "MyServiceName" -t
```

You can control an Apache service by its command line switches, too. To start an installed Apache service you'll use this:

```
httpd -k start
```

To stop an Apache service via the command line switches, use this:

```
httpd -k stop
```

or

```
httpd -k shutdown
```

You can also restart a running service and force it to reread its configuration file by using:

```
httpd -k restart
```

By default, all Apache services are registered to run as the system user (the LocalSystem account). The LocalSystem account has no privileges to your network via any Windows-secured mechanism, including the file system, named pipes, DCOM, or secure RPC. It has, however, wide privileges locally.

**Never grant any network privileges to the LocalSystem account! If you need Apache to be able to access network resources, create a separate account for Apache as noted below.**

You may want to create a separate account for running Apache service(s). Especially, if you have to access network resources via Apache, this is strongly recommended.

1. Create a normal domain user account, and be sure to memorize its password.
2. Grant the newly-created user a privilege of Log on as a service and Act as part of the operating system. On Windows NT 4.0 these privileges are granted via User Manager for Domains, but on Windows 2000 and XP you probably want to use Group Policy for propagating these settings. You can also manually set these via the Local Security Policy MMC snap-in.
3. Confirm that the created account is a member of the Users group.
4. Grant the account read and execute (RX) rights to all document and script folders (htdocs and cgi-bin for

example).

5. Grant the account change (RWXD) rights to the Apache logs directory.
6. Grant the account read and execute (RX) rights to the Apache .exe binary executable.

It is usually a good practice to grant the user the Apache service runs as read and execute (RX) access to the whole Apache2 directory, except the logs subdirectory, where the user has to have at least change (RWXD) rights.

If you allow the account to log in as a user and as a service, then you can log on with that account and test that the account has the privileges to execute the scripts, read the web pages, and that you can start Apache in a console window. If this works, and you have followed the steps above, Apache should execute as a service with no problems.

**Error code 2186** is a good indication that you need to review the "Log On As" configuration for the service, since Apache cannot access a required network resource. Also, pay close attention to the privileges of the user Apache is configured to run as.

When starting Apache as a service you may encounter an error message from the Windows Service Control Manager. For example, if you try to start Apache by using the Services applet in the Windows Control Panel, you may get the following message:

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

You will get this generic error if there is any problem with starting the Apache service. In order to see what is really causing the

problem you should follow the instructions for Running Apache for Windows from the Command Prompt.

There is some support for Apache on Windows 9x to behave in a similar manner as a service on Windows NT. It is **highly experimental**. It is not of production-class reliability, and its future is not guaranteed. It can be mostly regarded as a risky thing to play with - proceed with caution!

There are some differences between the two kinds of services you should be aware of:

- Apache will attempt to start and if successful it will run in the background. If you run the command

```
httpd -n "MyServiceName" -k start
```

via a shortcut on your desktop, for example, then if the service starts successfully, a console window will flash up but it immediately disappears. If Apache detects any errors on startup such as incorrect entries in the httpd.conf configuration file, the console window will remain visible. This will display an error message which will be useful in tracking down the cause of the problem.

- Windows 9x does not support NET START or NET STOP commands. You must control the Apache service on the command prompt via the -k switches.
- Apache and Windows 9x offer no support for running Apache as a specific user with network privileges. In fact, Windows 9x offers no security on the local machine, either. This is the simple reason because of which the Apache Software Foundation never endorses use of a Windows 9x -based system as a public Apache server. The primitive support for

Windows 9x exists only to assist the user in developing web content and learning the Apache server, and perhaps as an intranet server on a secured, private network.

Once you have confirmed that Apache runs correctly as a console application you can install, control and uninstall the pseudo-service with the same commands as on Windows NT. You can also use the Apache Service Monitor to manage Windows 9x pseudo-services.



## Running Apache as a Console Application

Running Apache as a service is usually the recommended way to use it, but it is sometimes easier to work from the command line (on Windows 9x running Apache from the command line is the recommended way due to the lack of reliable service support.)

To run Apache from the command line as a console application, use the following command:

```
httpd
```

Apache will execute, and will remain running until it is stopped by pressing Control-C.

You can also run Apache via the shortcut Start Apache in Console placed to Start Menu --> Programs --> Apache HTTP Server 2.0.xx --> Control Apache Server during the installation. This will open a console window and start Apache inside it. If you don't have Apache installed as a service, the window will remain visible until you stop Apache by pressing Control-C in the console window where Apache is running in. The server will exit in a few seconds. However, if you do have Apache installed as a service, the shortcut starts the service. If the Apache service is running already, the shortcut doesn't do anything.

You can tell a running Apache to stop by opening another console window and entering:

```
httpd -k shutdown
```

This should be preferred over pressing Control-C because this lets Apache end any current operations and clean up gracefully.

You can also tell Apache to restart. This forces it to reread the configuration file. Any operations in progress are allowed to

complete without interruption. To restart Apache, use:

```
httpd -k restart
```

Note for people familiar with the Unix version of Apache: these commands provide a Windows equivalent to `kill -TERM pid` and `kill -USR1 pid`. The command line option used, `-k`, was chosen as a reminder of the `kill` command used on Unix.

If the Apache console window closes immediately or unexpectedly after startup, open the Command Prompt from the Start Menu --> Programs. Change to the folder to which you installed Apache, type the command `apache`, and read the error message. Then change to the logs folder, and review the `error.log` file for configuration mistakes. If you accepted the defaults when you installed Apache, the commands would be:

```
c:  
cd "\\Program Files\\Apache Group\\Apache2\\bin"  
httpd
```

Then wait for Apache to stop, or press Control-C. Then enter the following:

```
cd ..\logs  
more < error.log
```

When working with Apache it is important to know how it will find the configuration file. You can specify a configuration file on the command line in two ways:

- `-f` specifies an absolute or relative path to a particular configuration file:

```
httpd -f "c:\my server files\anotherconfig.conf"
```

---

or

```
httpd -f files\anotherconfig.conf
```

- -n specifies the installed Apache service whose configuration file is to be used:

```
httpd -n "MyServiceName"
```

In both of these cases, the proper [ServerRoot](#) should be set in the configuration file.

If you don't specify a configuration file with -f or -n, Apache will use the file name compiled into the server, such as `conf\httpd.conf`. This built-in path is relative to the installation directory. You can verify the compiled file name from a value labelled as `SERVER_CONFIG_FILE` when invoking Apache with the -V switch, like this:

```
httpd -V
```

Apache will then try to determine its [ServerRoot](#) by trying the following, in this order:

1. A [ServerRoot](#) directive via the -C command line switch.
2. The -d switch on the command line.
3. Current working directory.
4. A registry entry which was created if you did a binary installation.
5. The server root compiled into the server. This is /apache by default, you can verify it by using `apache -V` and looking for

a value labelled as HTTPD\_ROOT.

During the installation, a version-specific registry key is created in the Windows registry. The location of this key depends on the type of the installation. If you chose to install Apache for all users, the key is located under the HKEY\_LOCAL\_MACHINE hive, like this (the version numbers will of course vary between different versions of Apache:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Group\Apache\2.0.43
```

Correspondingly, if you chose to install Apache for the current user only, the key is located under the HKEY\_CURRENT\_USER hive, the contents of which are dependent of the user currently logged on:

```
HKEY_CURRENT_USER\SOFTWARE\Apache Group\Apache\2.0.43
```

This key is compiled into the server and can enable you to test new versions without affecting the current version. Of course, you must take care not to install the new version in the same directory as another version.

If you did not do a binary install, Apache will in some scenarios complain about the missing registry key. This warning can be ignored if the server was otherwise able to find its configuration file.

The value of this key is the [ServerRoot](#) directory which contains the conf subdirectory. When Apache starts it reads the httpd.conf file from that directory. If this file contains a [ServerRoot](#) directive which contains a different directory from the one obtained from the registry key above, Apache will forget the registry key and use the directory from the configuration file. If you copy the Apache directory or configuration files to a new

location it is vital that you update the ServerRoot directive in the `httpd.conf` file to reflect the new location.



## Testing the installation

After starting Apache (either in a console window or as a service) it will be listening on port 80 (unless you changed the `Listen` directive in the configuration files or installed Apache only for the current user). To connect to the server and access the default page, launch a browser and enter this URL:

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

Apache should respond with a welcome page and a link to the Apache manual. If nothing happens or you get an error, look in the `error.log` file in the `logs` subdirectory. If your host is not connected to the net, or if you have serious problems with your DNS (Domain Name Service) configuration, you may have to use this URL:

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

If you happen to be running Apache on an alternate port, you need to explicitly put that in the URL:

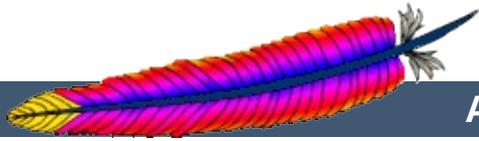
```
http://127.0.0.1:8080/
```

Once your basic installation is working, you should configure it properly by editing the files in the `conf` subdirectory. Again, if you change the configuration of the Windows NT service for Apache, first attempt to start it from the command line to make sure that the service starts with no errors.

Because Apache **cannot** share the same port with another TCP/IP application, you may need to stop, uninstall or reconfigure certain other services before running Apache. These conflicting services include other WWW servers and some firewall implementations.

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](#) > [Platform Specific Notes](#)

## Compiling Apache for Microsoft Windows

There are many important points before you begin compiling Apache. See [Using Apache with Microsoft Windows](#) before you begin.



## Requirements

Compiling Apache requires the following environment to be properly installed:

- Disk Space

Make sure you have at least 50 MB of free disk space available. After installation Apache requires approximately 10 MB of disk space, plus space for log and cache files, which can grow rapidly. The actual disk space requirements will vary considerably based on your chosen configuration and any third-party modules or libraries.

- Microsoft Visual C++ 5.0 or higher.

Apache can be built using the command line tools, or from within the Visual Studio IDE Workbench. The command line build requires the environment to reflect the PATH, INCLUDE, LIB and other variables that can be configured with the vcvars32 batch file:

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

- The Windows Platform SDK.

Visual C++ 5.0 builds require an updated Microsoft Windows Platform SDK to enable some Apache features. For command line builds, the Platform SDK environment is prepared by the setenv batch file:

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

The Platform SDK files distributed with Visual C++ 6.0 and later are sufficient, so users of later version may skip this requirement.

Note that the Windows Platform SDK update is required to enable all supported [mod\\_isapi](#) features. Without a recent update, Apache will issue warnings under MSVC++ 5.0 that some [mod\\_isapi](#) features will be disabled. Look for the update at <http://msdn.microsoft.com/downloads/sdks/platform/platform.>

- The awk utility (awk, gawk or similar).

To install Apache within the build system, several files are modified using the awk . exe utility. awk was chosen since it is a very small download (compared with Perl or WSH/VB) and accomplishes the task of generating files. Brian Kernighan's <http://cm.bell-labs.com/cm/cs/who/bwk/> site has a compiled native Win32 binary, <http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe> which you must save with the name awk . exe rather than awk95 . exe.

Note that Developer Studio IDE will only find awk . exe from the Tools menu Options... Directories tab (the Projects - VC++ Directories pane in Developer Studio 7.0) listing Executable file paths. Add the path for awk . exe to this list, and your system PATH environment variable, as needed.

Also note that if you are using Cygwin (<http://www.cygwin.com/>) the awk utility is named gawk . exe and that the file awk . exe is really a symlink to the gawk . exe file. The Windows command shell does not recognize symlinks, and because of that building InstallBin will fail. A workaround is to delete awk . exe from the cygwin installation and rename gawk . exe to awk . exe.

- [Optional] OpenSSL libraries (for [mod\\_ssl](#) and `ab.exe` with ssl support)

**Caution: there are significant restrictions and prohibitions on the use and distribution of strong cryptography and patented intellectual property throughout the world.** OpenSSL includes strong cryptography controlled by both export regulations and domestic law, as well as intellectual property protected by patent, in the United States and elsewhere. Neither the Apache Software Foundation nor the OpenSSL project can provide legal advice regarding possession, use, or distribution of the code provided by the OpenSSL project. **Consult your own legal counsel, you are responsible for your own actions.**

OpenSSL must be installed into a `src/lib` subdirectory named `openssl`, obtained from <http://www.openssl.org/source/>, in order to compile [mod\\_ssl](#) or the `abs` project (`ab.exe` with SSL support.) To prepare OpenSSL for both `release` and `debug` builds of Apache, and disable the patent protected features in OpenSSL, you might use the following build commands:

```
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.rel
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 -f makefile.rel
nmake -f makefile.dbg
```

Note; you can use the scripts in the `ms\` subdirectory, however, it's rather tricky to force `ms\do_masm.bat`, for example, to perform the patent encumbrances as mentioned above. Patches to add the `$*` argument list to the appropriate `.bat` lines in these scripts aren't incorporated, thus far.

- [Optional] zlib sources (for `mod_deflate`)

Zlib must be installed into a `src\lib` subdirectory named `zlib`, however those sources need not be compiled. The build system will compile the compression sources directly into the `mod_deflate` module. Zlib can be obtained from <http://www.zlib.net/> -- `mod_deflate` is confirmed to build correctly with version 1.1.4. To use a later version of zlib, upgrade to Apache HTTP Server release 2.2 or later.



First, unpack the Apache distribution into an appropriate directory. Open a command-line prompt and `cd` to that directory.

The master Apache makefile instructions are contained in the `Makefile.win` file. To compile Apache on Windows NT, simply use one of the following commands to compile the `release` or `debug` build, respectively:

```
nmake /f Makefile.win _apacher  
nmake /f Makefile.win _apached
```

Either command will compile Apache. The latter will include debugging information in the resulting files, making it easier to find bugs and track down problems.



---

Apache can also be compiled using VC++'s Visual Studio development environment. To simplify this process, a Visual Studio workspace, `Apache.dsw`, is provided. This workspace exposes the entire list of working `.dsp` projects that are required for the complete Apache binary release. It includes dependencies between the projects to assure that they are built in the appropriate order.

Open the `Apache.dsw` workspace, and select `InstallBin` (Release or Debug build, as desired) as the Active Project. `InstallBin` causes all related project to be built, and then invokes `Makefile.win` to move the compiled executables and dlls. You may personalize the `INSTDIR=` choice by changing `InstallBin`'s Settings, General tab, Build command line entry. `INSTDIR` defaults to the `/Apache2` directory. If you only want a test compile (without installing) you may build the `BuildBin` project instead.

The `.dsp` project files are distributed in Visual C++ 6.0 format. Visual C++ 5.0 (97) will recognize them. Visual C++ 7.0 (.net) must convert `Apache.dsw` plus the `.dsp` files into an `Apache.sln` plus `.msproj` files, be sure you reconvert the `.msproj` file if any of the source `.dsp` files change! This is really trivial, just open `Apache.dsw` in the VC++ 7.0 IDE once again.

Visual C++ 7.0 (.net) users should also use the Build menu, Configuration Manager dialog to uncheck both the Debug and Release Solution modules `abs`, `mod_ssl` and `mod_deflate`. These modules are built by invoking `nmake` or the IDE directly with the `BinBuild` target to build those modules explicitly, only if the `src\lib` directories `openssl` and/or `zlib` exist.

Exported `.mak` files pose a greater hassle, but they are required

for Visual C++ 5.0 users to build `mod_ssl`, `abs` (`ab` with SSL support) and/or `mod_deflate`. VC++ 7.0 (.net) users also benefit, nmake builds are faster than binenv builds. Build the entire project from within the VC++ 5.0 or 6.0 IDE, then use the Project Menu Export for all makefiles. You must build the projects first in order to create all dynamic auto-generated targets, so that dependencies can be parsed correctly. Run the following command to fix the paths so they will build anywhere:

```
perl src\lib\apr\build\fixwin32mak.pl
```

You must type this command from the *top level* directory of the httpd source tree. Every `.mak` and `.dep` project file within the current directory and below will be corrected, and the timestamps adjusted to reflect the `.dsp`.

If you contribute back a patch that revises project files, we must commit project files in Visual Studio 6.0 format. Changes should be simple, with minimal compilation and linkage flags that will be recognized by all VC++ 5.0 through 7.0 environments.



## Project Components

The Apache .dsw workspace and makefile.win nmake script both build the .dsp projects of the Apache server in the following sequence:

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

In addition, the modules\ subdirectory tree contains project files for the majority of the modules.

The support\ directory contains project files for additional programs that are not part of the Apache runtime, but are used by the administrator to test Apache and maintain password and log files. Windows-specific support projects are broken out in the support\win32\ directory.

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

Once Apache has been compiled, it needs to be installed in its server root directory. The default is the \Apache2 directory, of the same drive.

To build and install all the files into the desired folder *dir* automatically, use one of the following nmake commands:

```
nmake /f Makefile.win installr INSTDIR=dir
nmake /f Makefile.win installd INSTDIR=dir
```

The *dir* argument to INSTDIR gives the installation directory; it can be omitted if Apache is to be installed into \Apache2.

This will install the following:

- *dir*\bin\Apache.exe - Apache executable
- *dir*\bin\ApacheMonitor.exe - Service monitor taskbar icon utility
- *dir*\bin\htdigest.exe - Digest auth password file utility
- *dir*\bin\htdbm.exe - SDBM auth database password file utility
- *dir*\bin\htpasswd.exe - Basic auth password file utility
- *dir*\bin\logresolve.exe - Log file dns name lookup utility
- *dir*\bin\rotatelogs.exe - Log file cycling utility

- *dir\bin\wintty.exe* - Console window utility
- *dir\bin\libapr.dll* - Apache Portable Runtime shared library
- *dir\bin\libaprutil.dll* - Apache Utility Runtime shared library
- *dir\bin\libhttpd.dll* - Apache Core library
- *dir\modules\mod\_\*.so* - Loadable Apache modules
- *dir\conf* - Configuration directory
- *dir\logs* - Empty logging directory
- *dir\include* - C language header files
- *dir\lib* - Link library files

## **Warning about building Apache from the development tree**

Note only the .dsp files are maintained between release builds. The .mak files are NOT regenerated, due to the tremendous waste of reviewer's time. Therefore, you cannot rely on the NMAKE commands above to build revised .dsp project files unless you then export all .mak files yourself from the project. This is unnecessary if you build from within the Microsoft Developer Studio environment.

Also note it is very worthwhile to build the BuildBin target project (or the command line \_apacher or \_apached target) prior to exporting the make files. Many files are autogenerated in the build process. Only a full build provides all of the dependent files required to build proper dependency trees for correct build behavior.

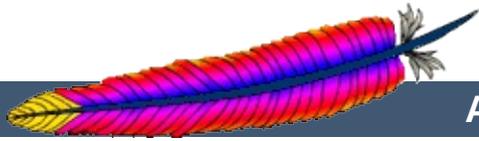
In order to create distribution .mak files, always review the generated .mak (or .dep) dependencies for Platform SDK or other garbage includes. The DevStudio\SharedIDE\bin\

(VC5) or DevStudio\Common\MSDev98\bin\ (VC6) directory contains the `sysincl.dat` file, which must list all exceptions. Update this file (including both forward and backslashed paths, such as both `sys/time.h` and `sys\time.h`) to include such dependencies. Including local-install paths in a distributed `.mak` file will cause the build to fail completely. And don't forget to run `srclib/apr/build/fixwin32mak.pl` in order to fix absolute paths within the `.mak` files.

---

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](#) > [Platform Specific Notes](#)

## Using Apache With Novell NetWare

This document explains how to install, configure and run Apache 2.0 under Novell NetWare 6.0 and above. If you find any bugs, or wish to contribute in other ways, please use our [bug reporting page](#).

The bug reporting page and dev-httpd mailing list are *not* provided to answer questions about configuration or running Apache. Before you submit a bug report or request, first consult this document, the [Frequently Asked Questions](#) page and the other relevant documentation topics. If you still have a question or problem, post it to the [novell.devsup.webserver](#) newsgroup, where many Apache users are more than willing to answer new and obscure questions about using Apache on NetWare.

Most of this document assumes that you are installing Apache from a binary distribution. If you want to compile Apache yourself (possibly to help with development, or to track down bugs), see the section on [Compiling Apache for NetWare](#) below.



## Requirements

Apache 2.0 is designed to run on NetWare 6.0 service pack 3 and above. If you are running a service pack less than SP3, you must install the latest [NetWare Libraries for C \(LibC\)](#).

NetWare service packs are available [here](#).

Apache 2.0 for NetWare can also be run in a NetWare 5.1 environment as long as the latest service pack or the latest version of the [NetWare Libraries for C \(LibC\)](#) has been installed .

**WARNING:** Apache 2.0 for NetWare has not been targeted for or tested in this environment.



## Connecting Apache to NetWare

Information on the latest version of Apache can be found on the Apache web server at <http://www.apache.org/>. This will list the current release, any more recent alpha or beta-test releases, together with details of mirror web and anonymous ftp sites. Binary builds of the latest releases of Apache 2.0 for NetWare can be downloaded from [here](#).



## Installing Apache on NetWare

There is no Apache install program for NetWare currently. If you are building Apache 2.0 for NetWare from source, you will need to copy the files over to the server manually.

Follow these steps to install Apache on NetWare from the binary download (assuming you will install to `sys:/apache2`):

- Unzip the binary download file to the root of the SYS: volume (may be installed to any volume)
- Edit the `httpd.conf` file setting `ServerRoot` and `ServerName` along with any file path values to reflect your correct server settings
- Add `SYS:/APACHE2` to the search path, for example:

```
SEARCH ADD SYS:\APACHE2
```

Follow these steps to install Apache on NetWare manually from your own build source (assuming you will install to `sys:/apache2`):

- Create a directory called Apache2 on a NetWare volume
- Copy `APACHE2.NLM`, `APRLIB.NLM` to `SYS:/APACHE2`
- Create a directory under `SYS:/APACHE2` called `BIN`
- Copy `HTDIGEST.NLM`, `HTPASSWD.NLM`, `HTDBM.NLM`, `LOGRES.NLM`, `ROTLOGS.NLM` to `SYS:/APACHE2/BIN`
- Create a directory under `SYS:/APACHE2` called `CONF`
- Copy the `HTTPD-STD.CONF` file to the `SYS:/APACHE2/CONF` directory and rename to `HTTPD.CONF`
- Copy the `MIME.TYPES`, `CHARSET.CONV` and `MAGIC` files to `SYS:/APACHE2/CONF` directory
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\ICONS` to `SYS:/APACHE2/ICONS`

- Copy all files and subdirectories in \HTTPD-2.0\DOCS\MANUAL to SYS:/APACHE2/MANUAL
- Copy all files and subdirectories in \HTTPD-2.0\DOCS\ERROR to SYS:/APACHE2/ERROR
- Copy all files and subdirectories in \HTTPD-2.0\DOCS\DOCR00T to SYS:/APACHE2/HTDOCS
- Create the directory SYS:/APACHE2/LOGS on the server
- Create the directory SYS:/APACHE2/CGI-BIN on the server
- Create the directory SYS:/APACHE2/MODULES and copy all nlm modules into the modules directory
- Edit the HTTPD.CONF file searching for all @@Value@@ markers and replacing them with the appropriate setting
- Add SYS:/APACHE2 to the search path, for example:

```
SEARCH ADD SYS:\APACHE2
```

Apache may be installed to other volumes besides the default SYS volume.

During the build process, adding the keyword "install" to the makefile command line will automatically produce a complete distribution package under the subdirectory DIST. Install Apache by simply copying the distribution that was produced by the makfiles to the root of a NetWare volume (see: [Compiling Apache for NetWare](#) below).



## Running Apache on NetWare

To start Apache just type `apache` at the console. This will load apache in the OS address space. If you prefer to load Apache in a protected address space you may specify the address space with the load statement as follows:

```
load address space = apache2 apache2
```

This will load Apache into an address space called `apache2`. Running multiple instances of Apache concurrently on NetWare is possible by loading each instance into its own protected address space.

After starting Apache, it will be listening to port 80 (unless you changed the `Listen` directive in the configuration files). To connect to the server and access the default page, launch a browser and enter the server's name or address. This should respond with a welcome page, and a link to the Apache manual. If nothing happens or you get an error, look in the `error_log` file in the `logs` directory.

Once your basic installation is working, you should configure it properly by editing the files in the `conf` directory.

To unload Apache running in the OS address space just type the following at the console:

```
unload apache2
```

or

```
apache2 shutdown
```

If apache is running in a protected address space specify the address space in the unload statement:

```
unload address space = apache2 apache2
```

When working with Apache it is important to know how it will find the configuration files. You can specify a configuration file on the command line in two ways:

- -f specifies a path to a particular configuration file

```
apache2 -f "vol:/my server/conf/my.conf"
```

```
apache -f test/test.conf
```

In these cases, the proper [ServerRoot](#) should be set in the configuration file.

If you don't specify a configuration file name with -f, Apache will use the file name compiled into the server, usually `conf/httpd.conf`. Invoking Apache with the -V switch will display this value labeled as `SERVER_CONFIG_FILE`. Apache will then determine its [ServerRoot](#) by trying the following, in this order:

- A `ServerRoot` directive via a -C switch.
- The -d switch on the command line.
- Current working directory
- The server root compiled into the server.

The server root compiled into the server is usually `sys:/apache2`. Invoking `apache` with the -V switch will display this value labeled as `HTTPD_ROOT`.

Apache 2.0 for NetWare includes a set of command line directives that can be used to modify or display information about the running instance of the web server. These directives are only

available while Apache is running. Each of these directives must be preceded by the keyword APACHE2.

## **RESTART**

Instructs Apache to terminate all running worker threads as they become idle, reread the configuration file and restart each worker thread based on the new configuration.

## **VERSION**

Displays version information about the currently running instance of Apache.

## **MODULES**

Displays a list of loaded modules both built-in and external.

## **DIRECTIVES**

Displays a list of all available directives.

## **SETTINGS**

Enables or disables the thread status display on the console. When enabled, the state of each running threads is displayed on the Apache console screen.

## **SHUTDOWN**

Terminates the running instance of the Apache web server.

## **HELP**

Describes each of the runtime directives.

By default these directives are issued against the instance of Apache running in the OS address space. To issue a directive against a specific instance running in a protected address space, include the -p parameter along with the name of the address space. For more information type "apache2 Help" on the command line.



## Configuring Apache for NetWare

Apache is configured by reading configuration files usually stored in the conf directory. These are the same as files used to configure the Unix version, but there are a few different directives for Apache on NetWare. See the [Apache documentation](#) for all the available directives.

The main differences in Apache for NetWare are:

- Because Apache for NetWare is multithreaded, it does not use a separate process for each request, as Apache does on some Unix implementations. Instead there are only threads running: a parent thread, and multiple child or worker threads which handle the requests.

Therefore the "process"-management directives are different:

MaxRequestsPerChild - Like the Unix directive, this controls how many requests a worker thread will serve before exiting. The recommended default, `MaxRequestsPerChild 0`, causes the thread to continue servicing request indefinitely. It is recommended on NetWare, unless there is some specific reason, that this directive always remain set to 0.

StartThreads - This directive tells the server how many threads it should start initially. The recommended default is `StartThreads 50`.

MinSpareThreads - This directive instructs the server to spawn additional worker threads if the number of idle threads ever falls below this value. The recommended default is `MinSpareThreads 10`.

MaxSpareThreads - This directive instructs the server to begin terminating worker threads if the number of idle threads

ever exceeds this value. The recommended default is `MaxSpareThreads 100`.

**[MaxThreads](#)** - This directive limits the total number of work threads to a maximum value. The recommended default is `ThreadsPerChild 250`.

**[ThreadStackSize](#)** - This directive tells the server what size of stack to use for the individual worker thread. The recommended default is `ThreadStackSize 65536`.

- The directives that accept filenames as arguments must use NetWare filenames instead of Unix names. However, because Apache uses Unix-style names internally, forward slashes must be used rather than backslashes. It is recommended that all rooted file paths begin with a volume name. If omitted, Apache will assume the `SYS:` volume which may not be correct.
- Apache for NetWare has the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the `\Apache2\modules` directory. To activate these, or other modules, the **[LoadModule](#)** directive must be used. For example, to activate the status module, use the following:

```
LoadModule status_module modules/status.nlm
```

Information on [creating loadable modules](#) is also available.

### **Additional NetWare specific directives:**

- **[CGIMapExtension](#)** - This directive maps a CGI file extension to a script interpreter.

- [SecureListen](#) - Enables SSL encryption for a specified port.
- [NWSSLTrustedCerts](#) - Adds trusted certificates that are used to create secure connections to proxied servers.
- [NWSSLUpgradeable](#) - Allow a connection created on the specified address/port to be upgraded to an SSL connection.



## Compiling Apache for NetWare

Compiling Apache requires MetroWerks CodeWarrior 6.x or higher. Once Apache has been built, it can be installed to the root of any NetWare volume. The default is the sys : /Apache2 directory.

Before running the server you must fill out the conf directory. Copy the file HTTPD-STD.CONF from the distribution conf directory and rename it to HTTPD.CONF. Edit the HTTPD.CONF file searching for all @@Value@@ markers and replacing them with the appropriate setting. Copy over the conf/magic and conf/mime.types files as well. Alternatively, a complete distribution can be built by including the keyword install when invoking the makefiles.

### Requirements:

The following development tools are required to build Apache 2.0 for NetWare:

- Metrowerks CodeWarrior 6.0 or higher with the [NetWare PDK 3.0](#) or higher.
- [NetWare Libraries for C \(LibC\)](#)
- [LDAP Libraries for C](#)
- [ZLIB Compression Library source code](#)
- AWK utility (awk, gawk or similar). AWK can be downloaded from <http://developer.novell.com/ndk/apache.htm>. The utility must be found in your windows path and must be named awk.exe.
- To build using the makefiles, you will need GNU make version 3.78.1 (GMake) available at <http://developer.novell.com/ndk/apache.htm>.

### Building Apache using the NetWare makefiles:

- Set the environment variable NOVELLIBC to the location of the NetWare Libraries for C SDK, for example:

```
Set NOVELLIBC=c:\novell\ndk\libc
```

- Set the environment variable METROWERKS to the location where you installed the Metrowerks CodeWarrior compiler, for example:

```
Set METROWERKS=C:\Program Files\Metrowerks\CodeWarrior
```

If you installed to the default location C:\Program Files\Metrowerks\CodeWarrior, you don't need to set this.

- Set the environment variable LDAPSDK to the location where you installed the LDAP Libraries for C, for example:

```
Set LDAPSDK=c:\Novell\NDK\cldap sdk\NetWare\libc
```

- Set the environment variable ZLIBSDK to the location where you installed the source code for the ZLib Library, for example:

```
Set ZLIBSDK=D:\NOVELL\zlib
```

- Set the environment variable AP\_WORK to the full path of the \httpd-2.0 directory.
- Set the environment variable APR\_WORK to the full path of the \httpd-2.0\src\lib\apr directory.
- Make sure that the path to the AWK utility and the GNU make utility (gmake.exe) have been included in the system's PATH environment variable.
- Download the source code and unzip to an appropriate

directory on your workstation.

- Change directory to `\httpd-2.0\src\lib\apr-util\uri` and build `GENURI.nlm` by running `"gmake -f nwgnumakefile"`.
- Copy the file `GENURI.nlm` to the `SYS:` volume of a NetWare server and run using the following command:

```
SYS:\genuri > sys:\uri_delims.h
```

- Copy the file `uri_delims.h` to the directory `\httpd-2.0\src\lib\apr-util\uri` on the build machine.
- Change directory to `\httpd-2.0\src\lib\apr` and build `APR` by running `"gmake -f nwgnumakefile"`
- Change directory to `\httpd-2.0\src\lib\pcre` and build `DFTABLES.nlm` by running `"gmake -f nwgnumakefile"`
- Change directory to `\httpd-2.0\server` and build `GENCHARS.nlm` by running `"gmake -f nwgnumakefile"`
- Copy the files `GENCHARS.nlm` and `DFTABLES.nlm` from their respective directories to the `SYS:` volume of a NetWare server and run them using the following commands:

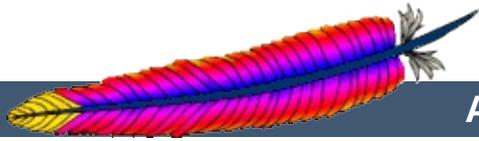
```
SYS:\genchars > sys:\test_char.h  
SYS:\dftables > sys:\chartables.c
```

- Copy the files `test_char.h` and `chartables.c` to the directory `\httpd-2.0\os\netware` on the build machine.
- Change directory to `\httpd-2.0` and build Apache by running `"gmake -f nwgnumakefile"`. You can create a distribution directory by adding an `install` parameter to the command, for example:

```
gmake -f nwgnumakefile install
```

## Additional make options

- `gmake -f nwgnmakefile`  
Builds release versions of all of the binaries and copies them to a `\release` destination directory.
- `gmake -f nwgnmakefile DEBUG=1`  
Builds debug versions of all of the binaries and copies them to a `\debug` destination directory.
- `gmake -f nwgnmakefile install`  
Creates a complete Apache distribution with binaries, docs and additional support files in a `\dist\Apache2` directory.
- `gmake -f nwgnmakefile installdev`  
Same as `install` but also creates a `\lib` and `\include` directory in the destination directory and copies headers and import files.
- `gmake -f nwgnmakefile clean`  
Cleans all object files and binaries from the `\release` or `\debug` build areas depending on whether `DEBUG` has been defined.
- `gmake -f nwgnmakefile clobber_all`  
Same as `clean` and also deletes the distribution directory if it exists.



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



## Apache HTTP Server Version 2.0

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

## Running a High-Performance Web Server on HPUX

Date: Wed, 05 Nov 1997 16:59:34 -0800  
From: Rick Jones <[raj@cup.hp.com](mailto:raj@cup.hp.com)>  
Reply-To: [raj@cup.hp.com](mailto:raj@cup.hp.com)  
Organization: Network Performance  
Subject: HP-UX tuning tips

Here are some tuning tips for HP-UX to add to the tuning page.

For HP-UX 9.X: Upgrade to 10.20

For HP-UX 10.[00|01|10]: Upgrade to 10.20

For HP-UX 10.20:

Install the latest cumulative ARPA Transport Patch. This will allow you to configure the size of the TCP connection lookup hash table. The default is 256 buckets and must be set to a power of two. This is accomplished with adb against the \*disc\* image of the kernel. The variable name is `tcp_hash_size`. Notice that it's critically important that you use "w" to write a 32 bit quantity, not "W" to write a 16 bit value when patching the disc image because the `tcp_hash_size` variable is a 32 bit quantity.

How to pick the value? Examine the output of <ftp://ftp.cup.hp.com/dist/networking/tools/connhist> and see how many total TCP connections exist on the system. You probably want that number divided by the hash table size to be reasonably small, say less than 10. Folks can look at HP's SPECweb96 disclosures for some common settings. These can be found at <http://www.specbench.org/>. If an HP-UX system was performing at 1000 SPECweb96 connections per second, the TIME\_WAIT time of 60 seconds would mean 60,000 TCP "connections" being tracked.

Folks can check their listen queue depths with <ftp://ftp.cup.hp.com/dist/networking/misc/listenq>.

If folks are running Apache on a PA-8000 based system, they should consider "chat'ing" the Apache executable to have a large page size. This would be "chatr +pi L <BINARY>". The GID of the running executable must have MLOCK privileges. Setprivgrp(1m) should be consulted for assigning MLOCK. The change can be validated by running Glance and examining the memory regions of the server(s) to make sure that they show a non-trivial fraction of the text segment being locked.

If folks are running Apache on MP systems, they might consider writing a small program that uses mpctl( ) to bind processes to processors. A simple pid % numcpu algorithm is probably sufficient. This might even go into the source code.

If folks are concerned about the number of FIN\_WAIT\_2 connections, they can use nettune to shrink the value of tcp\_keepstart. However, they should be careful there - certainly do not make it less than oh two to four minutes. If tcp\_hash\_size has been set well, it is probably OK to let the FIN\_WAIT\_2's take longer to timeout (perhaps even the default two hours) - they will not on average have a big impact on performance.

There are other things that could go into the code base, but that might be left for another email. Feel free to drop me a message if you or others are interested.

sincerely,

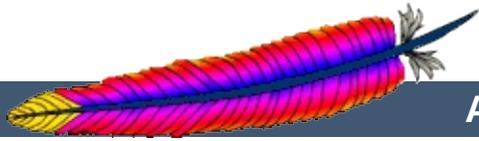
rick jones

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

---

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](#) > [Platform Specific Notes](#)

## The Apache EBCDIC Port

**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.



Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

(It is the SIEMENS family of mainframes running the [BS2000/OSD operating system](#). This mainframe OS nowadays features a SVR4-derived POSIX subsystem).

The port was started initially to

- prove the feasibility of porting [the Apache HTTP server](#) to this platform
- find a "worthy and capable" successor for the venerable [CERN-3.0](#) daemon (which was ported a couple of years ago), and to
- prove that Apache's preforking process model can on this platform easily outperform the accept-fork-serve model used by CERN by a factor of 5 or more.

This document serves as a rationale to describe some of the design decisions of the port to this machine.



One objective of the EBCDIC port was to maintain enough backwards compatibility with the (EBCDIC) CERN server to make the transition to the new server attractive and easy. This required the addition of a configurable method to define whether a HTML document was stored in ASCII (the only format accepted by the old server) or in EBCDIC (the native document format in the POSIX subsystem, and therefore the only realistic format in which the other POSIX tools like grep or sed could operate on the documents). The current solution to this is a "pseudo-MIME-format" which is intercepted and interpreted by the Apache server (see below). Future versions might solve the problem by defining an "ebcdic-handler" for all documents which must be converted.



Since all Apache input and output is based upon the BUFF data type and its methods, the easiest solution was to add the conversion to the BUFF handling routines. The conversion must be settable at any time, so a BUFF flag was added which defines whether a BUFF object has currently enabled conversion or not. This flag is modified at several points in the HTTP protocol:

- **set** before a request is received (because the request and the request header lines are always in ASCII format)
- **set/unset** when the request body is received - depending on the content type of the request body (because the request body may contain ASCII text or a binary file)
- **set** before a reply header is sent (because the response header lines are always in ASCII format)
- **set/unset** when the response body is sent - depending on the content type of the response body (because the response body may contain text or a binary file)



1. The relevant changes in the source are `#ifdef`'ed into two categories:

**`#ifdef CHARSET_EBCDIC`**

Code which is needed for any EBCDIC based machine. This includes character translations, differences in contiguity of the two character sets, flags which indicate which part of the HTTP protocol has to be converted and which part doesn't *etc.*

**`#ifdef _OSD_POSIX`**

Code which is needed for the SIEMENS BS2000/OSD mainframe platform only. This deals with include file differences and socket implementation topics which are only required on the BS2000/OSD platform.

2. The possibility to translate between ASCII and EBCDIC at the socket level (on BS2000 POSIX, there is a socket option which supports this) was intentionally *not* chosen, because the byte stream at the HTTP protocol level consists of a mixture of protocol related strings and non-protocol related raw file data. HTTP protocol strings are always encoded in ASCII (the GET request, any Header: lines, the chunking information *etc.*) whereas the file transfer parts (*i.e.*, GIF images, CGI output *etc.*) should usually be just "passed through" by the server. This separation between "protocol string" and "raw data" is reflected in the server code by functions like `bgets()` or `rvputs()` for strings, and functions like `bwrite()` for binary data. A global translation of everything would therefore be inadequate.

(In the case of text files of course, provisions must be made so that EBCDIC documents are always served in ASCII)

3. This port therefore features a built-in protocol level conversion for the server-internal strings (which the compiler translated to EBCDIC strings) and thus for all server-generated documents. The hard coded ASCII escapes `\012` and `\015` which are ubiquitous in the server code are an exception: they are already the binary encoding of the ASCII `\n` and `\r` and must not be converted to ASCII a second time. This exception is only relevant for server-generated strings; and *external* EBCDIC documents are not expected to contain ASCII newline characters.
4. By examining the call hierarchy for the BUFF management routines, I added an "ebcdic/ascii conversion layer" which would be crossed on every puts/write/get/gets, and a conversion flag which allowed enabling/disabling the conversions on-the-fly. Usually, a document crosses this layer twice from its origin source (a file or CGI output) to its destination (the requesting client): `file -> Apache`, and `Apache -> client`.

The server can now read the header lines of a CGI-script output in EBCDIC format, and then find out that the remainder of the script's output is in ASCII (like in the case of the output of a WWW Counter program: the document body contains a GIF image). All header processing is done in the native EBCDIC format; the server then determines, based on the type of document being served, whether the document body (except for the chunking information, of course) is in ASCII already or must be converted from EBCDIC.

5. For Text documents (MIME types `text/plain`, `text/html` etc.), an implicit translation to ASCII can be used, or (if the users prefer to store some documents in raw ASCII form for faster serving, or because the files reside on a NFS-mounted directory tree) can be served without conversion.

## Example:

to serve files with the suffix `.html` as a raw ASCII `text/html` document without implicit conversion (and suffix `.ascii` as ASCII `text/plain`), use the directives:

```
AddType text/x-ascii-html .html
AddType text/x-ascii-plain .ascii
```

Similarly, any `text/foo` MIME type can be served as "raw ASCII" by configuring a MIME type "`text/x-ascii-foo`" for it using `AddType`.

6. Non-text documents are always served "binary" without conversion. This seems to be the most sensible choice for, *e.g.*, GIF/ZIP/AU file types. This of course requires the user to copy them to the mainframe host using the "`rcp -b`" binary switch.
7. Server parsed files are always assumed to be in native (*i.e.*, EBCDIC) format as used on the machine, and are converted after processing.
8. For CGI output, the CGI script determines whether a conversion is needed or not: by setting the appropriate Content-Type, text files can be converted, or GIF output can be passed through unmodified. An example for the latter case is the `wwwcount` program which we ported as well.



## Binary Files

All files with a Content -Type: which does not start with text/ are regarded as *binary files* by the server and are not subject to any conversion. Examples for binary files are GIF images, gzip-compressed files and the like.

When exchanging binary files between the mainframe host and a Unix machine or Windows PC, be sure to use the ftp "binary" (TYPE I) command, or use the rcp -b command from the mainframe host (the -b switch is not supported in unix rcp's).

## Text Documents

The default assumption of the server is that Text Files (*i.e.*, all files whose Content -Type: starts with text/) are stored in the native character set of the host, EBCDIC.

## Server Side Included Documents

SSI documents must currently be stored in EBCDIC only. No provision is made to convert it from ASCII before processing.



Module	Status	Notes
<a href="#">core</a>	+	
<a href="#">mod_access</a>	+	
<a href="#">mod_actions</a>	+	
<a href="#">mod_alias</a>	+	
<a href="#">mod_asis</a>	+	
<a href="#">mod_auth</a>	+	
<a href="#">mod_auth_anon</a>	+	
<a href="#">mod_auth_dbm</a>	?	with own libdb.a
<a href="#">mod_autoindex</a>	+	
<a href="#">mod_cern_meta</a>	?	
<a href="#">mod_cgi</a>	+	
<a href="#">mod_digest</a>	+	
<a href="#">mod_dir</a>	+	
<a href="#">mod_so</a>	-	no shared libs
<a href="#">mod_env</a>	+	
<a href="#">mod_example</a>	-	(test bed only)
<a href="#">mod_expires</a>	+	
<a href="#">mod_headers</a>	+	
<a href="#">mod_imap</a>	+	
<a href="#">mod_include</a>	+	
<a href="#">mod_info</a>	+	
<a href="#">mod_log_agent</a>	+	
<a href="#">mod_log_config</a>	+	
<a href="#">mod_log_referer</a>	+	
<a href="#">mod_mime</a>	+	
<a href="#">mod_mime_magic</a>	?	not ported yet
<a href="#">mod_negotiation</a>	+	

<u>mod_proxy</u>	+	
<u>mod_rewrite</u>	+	untested
<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>	?	untested



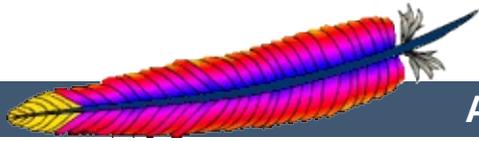
## Third Party Modules Status

Module	Status	Notes
<a href="#">mod_jserv</a>	-	JAVA still being ported.
<a href="#">mod_php3</a>	+	mod_php3 runs fine, with LDAP and GD and FreeType libraries.
<a href="#">mod_put</a>	?	untested
<a href="#">mod_session</a>	-	untested

---

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](#) > [Programs](#)

# httpd - Apache Hypertext Transfer Protocol Server

httpd is the Apache HyperText Transfer Protocol (HTTP) server program. It is designed to be run as a standalone daemon process. When used like this it will create a pool of child processes or threads to handle requests.

In general, httpd should not be invoked directly, but rather should be invoked via [apachectl](#) on Unix-based systems or [as a service on Windows NT, 2000 and XP](#) and [as a console application on Windows 9x and ME](#).

## See also

[Starting Apache](#)

[Stopping Apache](#)

[Configuration Files](#)

[Platform-specific Documentation](#)

[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 ]
```

On [Windows systems](#), the following additional arguments are available:

```
httpd [ -k install|config|uninstall ] [ -n name ]  
[ -w ]
```



## **-d *serverroot***

Set the initial value for the [ServerRoot](#) directive to *serverroot*. This can be overridden by the `ServerRoot` directive in the configuration file. The default is `/usr/local/apache2`.

## **-f *config***

Uses the directives in the file *config* on startup. If *config* does not begin with a `/`, then it is taken to be a path relative to the [ServerRoot](#). The default is `conf/httpd.conf`.

## **-k *start|restart|graceful|stop***

Signals `httpd` to start, restart, or stop. See [Stopping Apache](#) for more information.

## **-C *directive***

Process the configuration *directive* before reading config files.

## **-c *directive***

Process the configuration *directive* after reading config files.

## **-D *parameter***

Sets a configuration *parameter* which can be used with [<IfDefine>](#) sections in the configuration files to conditionally skip or process commands at server startup and restart.

## **-e *level***

Sets the [LogLevel](#) to *level* during server startup. This is useful for temporarily increasing the verbosity of the error messages to find problems during startup.

## **-E *file***

Send error messages during server startup to *file*.

## **-R *directory***

When the server is compiled using the `SHARED_CORE` rule,

this specifies the *directory* for the shared object files.

**-h**

Output a short summary of available command line options.

**-l**

Output a list of modules compiled into the server. This will **not** list dynamically loaded modules included using the [LoadModule](#) directive.

**-L**

Output a list of directives together with expected arguments and places where the directive is valid.

**-S**

Show the settings as parsed from the config file (currently only shows the virtualhost settings).

**-t**

Run syntax tests for configuration files only. The program immediately exits after these syntax parsing tests with either a return code of 0 (Syntax OK) or return code not equal to 0 (Syntax Error). If `-D DUMP_VHOSTS` is also set, details of the virtual host configuration will be printed.

**-v**

Print the version of httpd, and then exit.

**-V**

Print the version and build parameters of httpd, and then exit.

**-X**

Run httpd in debug mode. Only one worker will be started and the server will not detach from the console.

The following arguments are available only on the [Windows platform](#):

**-k `install|config|uninstall`**

Install Apache as a Windows NT service; change startup options for the Apache service; and uninstall the Apache service.

**-n *name***

The *name* of the Apache service to signal.

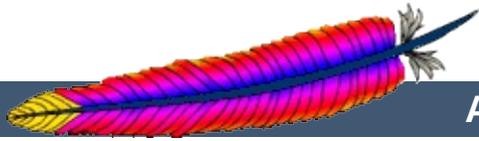
**-w**

Keep the console window open on error so that the error message can be read.

---

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](#) > [Programs](#)

## **ab - Apache HTTP server benchmarking tool**

ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.

### **See also**

[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***

Supply BASIC Authentication credentials to the server. The username and password are separated by a single : and sent on the wire base64 encoded. The string is sent regardless of whether the server needs it (*i.e.*, has sent an 401 authentication needed).

**-c *concurrency***

Number of multiple requests to perform at a time. Default is one request at a time.

**-C *cookie-name=value***

Add a Cookie: line to the request. The argument is typically in the form of a *name=value* pair. This field is repeatable.

**-d**

Do not display the "percentage served within XX [ms] table". (legacy support).

**-e *csv-file***

Write a Comma separated value (CSV) file which contains for each percentage (from 1% to 100%) the time (in milliseconds) it took to serve that percentage of the requests. This is usually more useful than the 'gnuplot' file; as the results are already 'binned'.

**-g *gnuplot-file***

Write all measured values out as a 'gnuplot' or TSV (Tab separate values) file. This file can easily be imported into packages like Gnuplot, IDL, Mathematica, Igor or even Excel. The labels are on the first line of the file.

**-h**

Display usage information.

**-H *custom-header***

Append extra headers to the request. The argument is

typically in the form of a valid header line, containing a colon-separated field-value pair (*i.e.*, "Accept-Encoding: zip/zop;8bit").

**-i**

Do HEAD requests instead of GET.

**-k**

Enable the HTTP KeepAlive feature, *i.e.*, perform multiple requests within one HTTP session. Default is no KeepAlive.

**-n *requests***

Number of requests to perform for the benchmarking session. The default is to just perform a single request which usually leads to non-representative benchmarking results.

**-p *POST-file***

File containing data to POST.

**-P *proxy-auth-username:password***

Supply BASIC Authentication credentials to a proxy en-route. The username and password are separated by a single : and sent on the wire base64 encoded. The string is sent regardless of whether the proxy needs it (*i.e.*, has sent an 407 proxy authentication needed).

**-q**

When processing more than 150 requests, ab outputs a progress count on `stderr` every 10% or 100 requests or so. The `-q` flag will suppress these messages.

**-s**

When compiled in (ab `-h` will show you) use the SSL protected `https` rather than the `http` protocol. This feature is experimental and very rudimentary. You probably do not want to use it.

**-S**

Do not display the median and standard deviation values, nor display the warning/error messages when the average and median are more than one or two times the standard deviation apart. And default to the min/avg/max values. (legacy support).

**-t *timelimit***

Maximum number of seconds to spend for benchmarking. This implies a -n 50000 internally. Use this to benchmark the server within a fixed total amount of time. Per default there is no timelimit.

**-T *content-type***

Content-type header to use for POST data.

**-v *verbosity***

Set verbosity level - 4 and above prints information on headers, 3 and above prints response codes (404, 200, etc.), 2 and above prints warnings and info.

**-V**

Display version number and exit.

**-w**

Print out results in HTML tables. Default table is two columns wide, with a white background.

**-x *<table>-attributes***

String to use as attributes for <table>. Attributes are inserted <table *here* >.

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

Use a proxy server for the requests.

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

String to use as attributes for <tr>.

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

String to use as attributes for <td>.



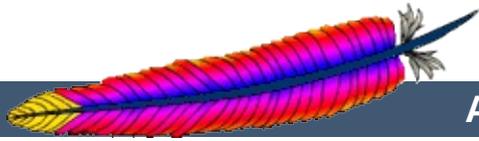
There are various statically declared buffers of fixed length. Combined with the lazy parsing of the command line arguments, the response headers from the server and other external inputs, this might bite you.

It does not implement HTTP/1.x fully; only accepts some 'expected' forms of responses. The rather heavy use of `strstr(3)` shows up top in profile, which might indicate a performance problem; *i.e.*, you would measure the ab performance rather than the server's.

---

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](#) > [Programs](#)

# apachectl - Apache HTTP Server Control Interface

`apachectl` is a front end to the Apache HyperText Transfer Protocol (HTTP) server. It is designed to help the administrator control the functioning of the Apache [httpd](#) daemon.

The `apachectl` script can operate in two modes. First, it can act as a simple front-end to the [httpd](#) command that simply sets any necessary environment variables and then invokes [httpd](#), passing through any command line arguments. Second, `apachectl` can act as a SysV init script, taking simple one-word arguments like `start`, `restart`, and `stop`, and translating them into appropriate signals to [httpd](#).

If your Apache installation uses non-standard paths, you will need to edit the `apachectl` script to set the appropriate paths to the [httpd](#) binary. You can also specify any necessary [httpd](#) command line arguments. See the comments in the script for details.

The `apachectl` script returns a 0 exit value on success, and >0 if an error occurs. For more details, view the comments in the script.

## See also

- [Starting Apache](#)
- [Stopping Apache](#)
- [Configuration Files](#)
- [Platform Docs](#)
- [httpd](#)



## Synopsis

When acting in pass-through mode, `apachectl` can take all the arguments available for the [httpd](#) binary.

**apachectl** [ *httpd-argument* ]

When acting in SysV init mode, `apachectl` takes simple, one-word commands, defined below.

**apachectl** *command*



Only the SysV init-style options are defined here. Other arguments are defined on the [httpd](#) manual page.

### **start**

Start the Apache [httpd](#) daemon. Gives an error if it is already running. This is equivalent to `apachectl -k start`.

### **stop**

Stops the Apache [httpd](#) daemon. This is equivalent to `apachectl -k stop`.

### **restart**

Restarts the Apache [httpd](#) daemon. If the daemon is not running, it is started. This command automatically checks the configuration files as in `configtest` before initiating the restart to make sure the daemon doesn't die. This is equivalent to `apachectl -k restart`.

### **fullstatus**

Displays a full status report from [mod\\_status](#). For this to work, you need to have [mod\\_status](#) enabled on your server and a text-based browser such as `Lynx` available on your system. The URL used to access the status report can be set by editing the `STATUSURL` variable in the script.

### **status**

Displays a brief status report. Similar to the `fullstatus` option, except that the list of requests currently being served is omitted.

### **graceful**

Gracefully restarts the Apache [httpd](#) daemon. If the daemon is not running, it is started. This differs from a normal restart in that currently open connections are not aborted. A side effect

is that old log files will not be closed immediately. This means that if used in a log rotation script, a substantial delay may be necessary to ensure that the old log files are closed before processing them. This command automatically checks the configuration files as in `configtest` before initiating the restart to make sure Apache doesn't die. This is equivalent to `apachectl -k graceful`.

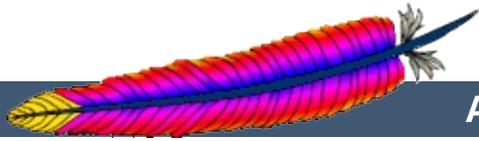
### **configtest**

Run a configuration file syntax test. It parses the configuration files and either reports `Syntax Ok` or detailed information about the particular syntax error. This is equivalent to `apachectl -t`.

The following additional option is available, but deprecated.

### **startssl**

This is equivalent to `apachectl -k start -DSSL`. We recommend that you use that command explicitly, or you adjust your `httpd.conf` to remove the `<IfDefine>` section so that SSL will always be available.



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



## Apache HTTP Server Version 2.0

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

## apxs - APache eXtenSion tool

apxs is a tool for building and installing extension modules for the Apache HyperText Transfer Protocol (HTTP) server. This is achieved by building a dynamic shared object (DSO) from one or more source or object *files* which then can be loaded into the Apache server under runtime via the [LoadModule](#) directive from [mod\\_so](#).

So to use this extension mechanism your platform has to support the DSO feature and your Apache [httpd](#) binary has to be built with the [mod\\_so](#) module. The apxs tool automatically complains if this is not the case. You can check this yourself by manually running the command

```
$ httpd -l
```

The module [mod\\_so](#) should be part of the displayed list. If these requirements are fulfilled you can easily extend your Apache server's functionality by installing your own modules with the DSO mechanism by the help of this apxs tool:

```
$ 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
$ _
```

The arguments *files* can be any C source file (.c), a object file (.o) or even a library archive (.a). The apxs tool automatically recognizes

these extensions and automatically used the C source files for compilation while just using the object and archive files for the linking phase. But when using such pre-compiled objects make sure they are compiled for position independent code (PIC) to be able to use them for a dynamically loaded shared object. For instance with GCC you always just have to use `-fpic`. For other C compilers consult its manual page or at watch for the flags `apxs` uses to compile the object files.

For more details about DSO support in Apache read the documentation of [mod\\_so](#) or perhaps even read the `src/modules/standard/mod_so.c` source file.

## See also

[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* ...



## Common Options

### **-n *modname***

This explicitly sets the module name for the `-i` (install) and `-g` (template generation) option. Use this to explicitly specify the module name. For option `-g` this is required, for option `-i` the `apxs` tool tries to determine the name from the source or (as a fallback) at least by guessing it from the filename.

## Query Options

### **-q**

Performs a query for `apxs`'s knowledge about certain settings. The *query* parameters can be one or more of the following strings: `CC`, `CFLAGS`, `CFLAGS_SHLIB`, `INCLUDEDIR`, `LD_SHLIB`, `LD_FLAGS_SHLIB`, `LIBEXECDIR`, `LIBS_SHLIB`, `SBINDIR`, `SYSCONFDIR`, `TARGET`. Use this for manually determining settings. For instance use

```
INC=-I`apxs -q INCLUDEDIR`
```

inside your own Makefiles if you need manual access to Apache's C header files.

## Configuration Options

### **-S *name=value***

This option changes the `apxs` settings described above.

## Template Generation Options

### **-g**

This generates a subdirectory *name* (see option `-n`) and there two files: A sample module source file named `mod_name.c`

which can be used as a template for creating your own modules or as a quick start for playing with the apxs mechanism. And a corresponding Makefile for even easier build and installing of this module.

## DSO Compilation Options

### **-c**

This indicates the compilation operation. It first compiles the C source files (.c) of *files* into corresponding object files (.o) and then builds a dynamically shared object in *dsofile* by linking these object files plus the remaining object files (.o and .a) of *files*. If no -o option is specified the output file is guessed from the first filename in *files* and thus usually defaults to `mod_name.so`.

### **-o *dsofile***

Explicitly specifies the filename of the created dynamically shared object. If not specified and the name cannot be guessed from the *files* list, the fallback name `mod_unknown.so` is used.

### **-D *name=value***

This option is directly passed through to the compilation command(s). Use this to add your own defines to the build process.

### **-I *incdir***

This option is directly passed through to the compilation command(s). Use this to add your own include directories to search to the build process.

### **-L *libdir***

This option is directly passed through to the linker command. Use this to add your own library directories to search to the build process.

**-l *libname***

This option is directly passed through to the linker command. Use this to add your own libraries to search to the build process.

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

This option passes *compiler-flags* as additional flags to the compiler command. Use this to add local compiler-specific options.

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

This option passes *linker-flags* as additional flags to the linker command. Use this to add local linker-specific options.

## DSO Installation and Configuration Options

**-i**

This indicates the installation operation and installs one or more dynamically shared objects into the server's *modules* directory.

**-a**

This activates the module by automatically adding a corresponding [LoadModule](#) line to Apache's `httpd.conf` configuration file, or by enabling it if it already exists.

**-A**

Same as option -a but the created [LoadModule](#) directive is prefixed with a hash sign (`#`), *i.e.*, the module is just prepared for later activation but initially disabled.

**-e**

This indicates the editing operation, which can be used with the -a and -A options similarly to the -i operation to edit Apache's `httpd.conf` configuration file without attempting to install the module.



Assume you have an Apache module named `mod_foo.c` available which should extend Apache's server functionality. To accomplish this you first have to compile the C source into a shared object suitable for loading into the Apache server under runtime via the following command:

```
$ 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
$ _
```

Then you have to update the Apache configuration by making sure a `LoadModule` directive is present to load this shared object. To simplify this step `apxs` provides an automatic way to install the shared object in its "modules" directory and updating the `httpd.conf` file accordingly. This can be achieved by running:

```
$ 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]
$ _
```

This way a line named

```
LoadModule foo_module modules/mod_foo.so
```

is added to the configuration file if still not present. If you want to have this disabled per default use the `-A` option, *i.e.*

```
$ apxs -i -A mod_foo.c
```

For a quick test of the `apxs` mechanism you can create a sample Apache module template plus a corresponding Makefile via:

---

```
$ apxs -g -n foo
Creating [DIR] foo
Creating [FILE] foo/Makefile
Creating [FILE] foo/mod_foo.c
$ _
```

Then you can immediately compile this sample module into a shared object and load it into the Apache server:

```
$ 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
$ _
```

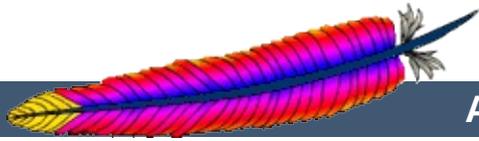
You can even use apxs to compile complex modules outside the Apache source tree, like PHP3:

```
$ 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
$ _
```

because apxs automatically recognized C source files and object files. Only C source files are compiled while remaining object files are used for the linking phase.

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](#) > [Programs](#)

## configure - Configure the source tree

The `configure` script configures the source tree for compiling and installing the Apache HTTP Server on your particular platform. Various options allow the compilation of a server corresponding to your personal requirements.

This script, included in the root directory of the source distribution, is for compilation on Unix and Unix-like systems only. For other platforms, see the [platform](#) documentation.

### See also

[Compiling and Installing](#)



You should call the `configure` script from within the root directory of the distribution.

```
./configure [OPTION]... [VAR=VALUE]...
```

To assign environment variables (e.g. `CC`, `CFLAGS` ...), specify them as `VAR=VALUE`. See [below](#) for descriptions of some of the useful variables.



- [Configuration options](#)
- [Installation directories](#)
- [System types](#)
- [Optional features](#)
- [Options for support programs](#)

## Configuration options

The following options influence the behavior of `configure` itself.

**-C**

**--config-cache**

This is an alias for `--cache-file=config.cache`

**--cache-file=FILE**

The test results will be cached in file *FILE*. This option is disabled by default.

**-h**

**--help [short|recursive]**

Output the help and exit. With the argument `short` only options specific to this package will displayed. The argument `recursive` displays the short help of all the included packages.

**-n**

**--no-create**

The `configure` script is run normally but does not create output files. This is useful to check the test results before generating makefiles for compilation.

**-q**

**--quiet**

Do not print checking . . . messages during the `configure`

process.

**--srcdir=DIR**

Defines directory *DIR* to be the source file directory. Default is the directory, where configure is located, or the parent directory . . .

**--silent**

Same as --quiet

**-V**

**--version**

Display copyright information and exit.

## Installation directories

These options define the installation directory. The installation tree depends on the selected layout.

**--prefix=PREFIX**

Install architecture-independent files in *PREFIX*. By default the installation directory is set to `/usr/local/apache2`.

**--exec-prefix=EPREFIX**

Install architecture-dependent files in *EPREFIX*. By default the installation directory is set to the *PREFIX* directory.

By default, `make install` will install all the files in `/usr/local/apache2/bin`, `/usr/local/apache2/lib` etc. You can specify an installation prefix other than `/usr/local/apache2` using `--prefix`, for instance `--prefix=$HOME`.

## Define a directory layout

**--enable-layout=LAYOUT**

Configure the source code and build scripts to assume an

installation tree based on the layout *LAYOUT*. This allows you to separately specify the locations for each type of file within the Apache HTTP Server installation. The `config.layout` file contains several example configurations, and you can also create your own custom configuration following the examples. The different layouts in this file are grouped into `<Layout F00> . . . </Layout>` sections and referred to by name as in `F00`. The default layout is Apache.

### **Fine tuning of the installation directories**

For better control of the installation directories, use the options below. Please note that the directory defaults are set by `autoconf` and be overwritten by the corresponding layout setting.

#### **--bindir=*DIR***

Install user executables in *DIR*. The user executables are supporting programs like [htpasswd](#), [dbmmanage](#), etc. which are useful for site administrators. By default *DIR* is set to `EPREFIX/bin`.

#### **--datadir=*DIR***

Install read-only architecture-independent data in *DIR*. By default `datadir` is set to `PREFIX/share`. This option is offered by `autoconf` and currently unused.

#### **--includedir=*DIR***

Install C header files in *DIR*. By default `includedir` is set to `EPREFIX/include`.

#### **--infodir=*DIR***

Install info documentation in *DIR*. By default `infodir` is set to `PREFIX/info`. This option is currently unused.

#### **--libdir=*DIR***

Install object code libraries in *DIR*. By default `libdir` is set to `EPREFIX/lib`.

**--libexecdir=DIR**

Install the program executables (i.e., shared modules) in *DIR*. By default `libexecdir` is set to *EPREFIX/libexec*.

**--localstatedir=DIR**

Install modifiable single-machine data in *DIR*. By default `localstatedir` is set to *PREFIX/var*. This option is offered by `autoconf` and currently unused.

**--mandir=DIR**

Install the man documentation in *DIR*. By default `mandir` is set to *EPREFIX/man*.

**--oldincludedir=DIR**

Install C header files for non-gcc in *DIR*. By default `oldincludedir` is set to `/usr/include`. This option is offered by `autoconf` and currently unused.

**--sbindir=DIR**

Install the system administrator executables in *DIR*. Those are server programs like [httpd](#), [apachectl](#), [suexec](#), etc. which are necessary to run the Apache HTTP Server. By default `sbindir` is set to *EPREFIX/sbin*.

**--sharedstatedir=DIR**

Install modifiable architecture-independent data in *DIR*. By default `sharedstatedir` is set to *PREFIX/com*. This option is offered by `autoconf` and currently unused.

**--sysconfdir=DIR**

Install read-only single-machine data like the server configuration files `httpd.conf`, `mime.types`, etc. in *DIR*. By default `sysconfdir` is set to *PREFIX/conf*.

## System types

These options are used to cross-compile the Apache HTTP Server

to run on another system. In normal cases, when building and running the server on the same system, these options are not used.

**--build=*BUILD***

Defines the system type of the system on which the tools are being built. It defaults to the result of the script `config.guess`.

**--host=*HOST***

Defines the system type of the system on which the server will run. *HOST* defaults to *BUILD*.

**--target=*TARGET***

Configure for building compilers for the system type *TARGET*. It defaults to *HOST*. This option is offered by `autoconf` and not necessary for the Apache HTTP Server.

## Optional Features

These options are used to fine tune the features your HTTP server will have.

### General syntax

Generally you can use the following syntax to enable or disable a feature:

**--disable-*FEATURE***

Do not include *FEATURE*. This is the same as `--enable-FEATURE=no`.

**--enable-*FEATURE*[=*ARG*]**

Include *FEATURE*. The default value for *ARG* is yes.

**--enable-*MODULE*=shared**

The corresponding module will be build as DSO module.

**--enable-*MODULE*=static**

By default enabled modules are linked statically. You can force this explicitly.

### Note

`configure` will not complain about `--enable-foo` even if `foo` doesn't exist, so you need to type carefully.

### Modules enabled by default

Some modules are compiled by default and have to be disabled explicitly. Use the following options to remove discrete modules from the compilation process.

#### **--disable-actions**

Disable action triggering on requests, which is provided by [mod\\_actions](#).

#### **--disable-alias**

Disable the mapping of requests to different parts of the filesystem, which is provided by [mod\\_alias](#).

#### **--disable-asis**

Disable support for as-is filetypes, which is provided by [mod\\_asis](#).

#### **--disable-auth**

Disable user-based access control provided by [mod\\_auth](#). This module provides for HTTP Basic Authentication, where the usernames and passwords are stored in plain text files.

#### **--disable-autoindex**

Disable the directory listing functionality provided by [mod\\_autoindex](#).

#### **--disable-access**

Disable host-based access control provided by [mod\\_access](#).

#### **--disable-cgi**

[mod\\_cgi](#), which provides support for CGI scripts, is enabled by default when using a non-threaded MPM. Use this option to disable CGI support.

### **--disable-cgid**

When using the threaded MPMs [worker](#) or [perchild](#) support for CGI scripts is provided by [mod\\_cgid](#) by default. To disable CGI support use this option.

### **--disable-charset-lite**

Disable character set translation provided by [mod\\_charset\\_lite](#). This module will be installed by default only on EBCDIC systems.

### **--disable-dir**

Disable directory request handling provided by [mod\\_dir](#).

### **--disable-env**

Disable setting and clearing of environment variables, which is provided by [mod\\_env](#).

### **--disable-http**

Disable the HTTP protocol handling. The `http` module is a basic one, enabling the server to function as an HTTP server. It is only useful to disable it if you want to use another protocol module instead. **Don't disable this module unless you are really sure what you are doing.**

Note: This module will always be linked statically.

### **--disable-imap**

Disable support for server based imagemaps, which provided by [mod\\_imap](#).

### **--disable-include**

Disable Server Side Includes provided by [mod\\_include](#).

### **--disable-log-config**

Disable the logging configuration provided by

[mod\\_log\\_config](#). You won't be able to log requests to the server without this module.

**--disable-mime**

[mod\\_mime](#) associates the requested filename's extensions with the file's behavior and content (mime-type, language, character set and encoding). Disabling the mapping of file-extensions to MIME is normally not recommended.

**--disable-negotiation**

Disable content negotiation provided by [mod\\_negotiation](#).

**--disable-setenvif**

Disable support for basing environment variables on headers, which is provided by [mod\\_setenvif](#).

**--disable-status**

Disable the process/thread monitoring, which is provided by [mod\\_status](#).

**--disable-userdir**

Disable the mapping of requests to user-specific directories, which is provided by [mod\\_userdir](#).

**Modules, disabled by default**

Some modules are compiled by default and have to be enabled explicitly or by using the keywords `most` or `all` (see `--enable-mods-shared` below for further explanation) to be available.

Therefore use the options below.

**--enable-auth-anon**

Enable anonymous user access provided by [mod\\_auth\\_anon](#).

**--enable-auth-dbm**

[mod\\_auth\\_dbm](#) provides for HTTP Basic Authentication, where the usernames and passwords are stored in DBM type

database files. Use this option to enable the module.

**--enable-auth-digest**

Enable RFC2617 Digest authentication provided by [mod\\_auth\\_digest](#). This module uses plain text files to store the credentials.

**--enable-auth-ldap**

Enable LDAP based authentication provided by [mod\\_auth\\_ldap](#).

**--enable-cache**

Enable dynamic file caching provided by [mod\\_cache](#). This experimental module may be interesting for servers with high load or caching proxy servers. At least one storage management module (e.g. [mod\\_disk\\_cache](#) or [mod\\_mem\\_cache](#)) is also necessary.

**--enable-cern-meta**

Enable the CERN-type meta files support provided by [mod\\_cern\\_meta](#).

**--enable-charset-lite**

Enable character set translation provided by [mod\\_charset\\_lite](#). This module will be installed by default only on EBCDIC systems. On other systems, you have to enable it.

**--enable-dav**

Enable the WebDAV protocol handling provided by [mod\\_dav](#). Support for filesystem resources is provided by the separate module [mod\\_dav\\_fs](#). This module is also automatically enabled with --enable-dav.

Note: [mod\\_dav](#) can only be used together with the http protocol module.

**--enable-dav-fs**

Enable DAV support for filesystem resources, which is

provided by [mod\\_dav\\_fs](#). This module is a provider for the [mod\\_dav](#) module, so you should also use `--enable-dav`.

**--enable-deflate**

Enable deflate transfer encoding provided by [mod\\_deflate](#).

**--enable-disk-cache**

Enable disk caching provided by [mod\\_disk\\_cache](#).

**--enable-expires**

Enable Expires header control provided by [mod\\_expires](#).

**--enable-ext-filter**

Enable the external filter support provided by [mod\\_ext\\_filter](#).

**--enable-file-cache**

Enable the file cache provided by [mod\\_file\\_cache](#).

**--enable-headers**

Enable control of HTTP headers provided by [mod\\_headers](#).

**--enable-info**

Enable the server information provided by [mod\\_info](#).

**--enable-ldap**

Enable LDAP caching and connection pooling services provided by [mod\\_ldap](#).

**--enable-logio**

Enable logging of input and output bytes including headers provided by [mod\\_logio](#).

**--enable-mem-cache**

Enable memory caching provided by [mod\\_mem\\_cache](#).

**--enable-mime-magic**

Enable automatic determining of MIME types, which is provided by [mod\\_mime\\_magic](#).

**--enable-isapi**

Enable the isapi extension support provided by [mod\\_isapi](#).

### **--enable-proxy**

Enable the proxy/gateway functionality provided by [mod\\_proxy](#). The proxying capabilities for CONNECT, FTP and HTTP are provided by the separate modules [mod\\_proxy\\_connect](#), [mod\\_proxy\\_ftp](#) and [mod\\_proxy\\_http](#). These three modules are also automatically enabled with --enable-proxy.

### **--enable-proxy-connect**

Enable proxy support for CONNECT request handling, which is provided by [mod\\_proxy\\_connect](#). This module is an extension for the [mod\\_proxy](#) module, so you should also use --enable-proxy.

### **--enable-proxy-ftp**

Enable proxy support for FTP requests, which is provided by [mod\\_proxy\\_ftp](#). This module is an extension for the [mod\\_proxy](#) module, so you should also use --enable-proxy.

### **--enable-proxy-http**

Enable proxy support for HTTP requests, which is provided by [mod\\_proxy\\_http](#). This module is an extension for the [mod\\_proxy](#) module, so you should also use --enable-proxy.

### **--enable-rewrite**

Enable rule based URL manipulation provided by [mod\\_rewrite](#).

### **--enable-so**

Enable DSO capability provided by [mod\\_so](#). This module will be automatically enabled if you use the --enable-mods-shared option.

**--enable-speling**

Enable the functionality to correct common URL misspellings, which is provided by [mod\\_speling](#).

**--enable-ssl**

Enable support for SSL/TLS provided by [mod\\_ssl](#).

**--enable-unique-id**

Enable the generation of per-request unique ids, which is provided by [mod\\_unique\\_id](#).

**--enable-usertrack**

Enable user-session tracking provided by [mod\\_usertrack](#).

**--enable-vhost-alias**

Enable mass virtual hosting provided by [mod\\_vhost\\_alias](#).

### Modules for developers

The following modules are useful only for developers and testing purposes and are disabled by default. Use the following options to enable them. If you are not sure whether you need one of these modules, omit them.

**--enable-bucketeer**

Enable the manipulation filter for buckets, which is provided by `mod_bucketeer`.

**--enable-case-filter**

Enable the example uppercase conversion output filter support of `mod_case_filter`.

**--enable-case-filter-in**

Enable the example uppercase conversion input filter support of `mod_case_filter_in`.

**--enable-echo**

Enable the ECHO server provided by [mod\\_echo](#).

**--enable-example**

Enable the example and demo module [mod\\_example](#).

**--enable-optional-fn-export**

Enable the example for an optional function exporter, which is provided by `mod_optional_fn_export`.

**--enable-optional-fn-import**

Enable the example for an optional function importer, which is provided by `mod_optional_fn_import`.

**--enable-optional-hook-export**

Enable the example for an optional hook exporter, which is provided by `mod_optional_hook_export`.

**--enable-optional-hook-import**

Enable the example optional hook importer, which is provided by `mod_optional_hook_import`.

## MPMs and third-party modules

To add the necessary Multi Processing Module and additional third-party modules use the following options:

**--with-module=*module-type:module-file*[, *module-type:module-file*]**

Add one or more third-party modules to the list of statically linked modules. The module source file *module-file* will be searched in the `modules/module-type` subdirectory of your Apache HTTP server source tree. If it is not found there `configure` is considering *module-file* to be an absolute file path and tries to copy the source file into the *module-type* subdirectory. If the subdirectory doesn't exist it will be created and populated with a standard `Makefile.in`.

This option is useful to add small external modules consisting of one source file. For more complex modules you should read the vendor's documentation.

## Note

If you want to build a DSO module instead of a statically linked use [apxs](#).

### **--with-mpm=MPM**

Choose the process model for your server. You have to select exactly one [Multi-Processing Module](#). Otherwise the [default MPM](#) for your operating system will be taken. Possible MPMs are [beos](#), [leader](#), [mpmt\\_os2](#), [perchild](#), [prefork](#), [threadpool](#) and [worker](#).

## Cumulative and other options

### **--enable-maintainer-mode**

Turn on debugging and compile time warnings.

### **--enable-mods-shared=MODULE-LIST**

Defines a list of modules to be enabled and build as dynamic shared modules. This mean, these module have to be loaded dynamically by using the [LoadModule](#) directive.

*MODULE-LIST* is a space separated list of modulenames enclosed by quotation marks. The module names are given without the preceding `mod_`. For example:

```
--enable-mods-shared='headers rewrite dav'
```

Additionally you can use the special keywords `all` and `most`. For example,

```
--enable-mods-shared=most
```

will compile most modules and build them as DSO modules.

### **--enable-modules=MODULE-LIST**

This option behaves similar to `--enable-mods-shared`, but will link the given modules statically. This means, these modules will always be present while running `httpd`. They need not be loaded with `LoadModule`.

**`--enable-v4-mapped`**

Allow IPv6 sockets to handle IPv4 connections.

**`--with-port=PORT`**

This defines the port on which `httpd` will listen. This port number is used when generating the configuration file `httpd.conf`. The default is 80.

**`--with-program-name`**

Define an alternative executable name. The default is `httpd`.

## Optional packages

These options are used to define optional packages.

### General syntax

Generally you can use the following syntax to define an optional package:

**`--with-PACKAGE[=ARG]`**

Use the package `PACKAGE`. The default value for `ARG` is `yes`.

**`--without-PACKAGE`**

Do not use the package `PACKAGE`. This is the same as `--with-PACKAGE=no`. This option is provided by `autoconf` but not very useful for the Apache HTTP Server.

### Specific packages

**`--with-apr=DIR|FILE`**

The Apache Portable Runtime (APR) is part of the `httpd`

source distribution and will automatically be build together with the HTTP server. If you want to use an already installed APR instead you have to tell configure the path to the `apr-config` script. You may set the absolute path and name or the directory to the installed APR. `apr-config` must exist within this directory or the subdirectory `bin`.

**--with-apr-util=DIR|FILE**

The Apache Portable Runtime Utilities (APU) are part of the `httpd` source distribution and will automatically be build together with the HTTP server. If you want to use an already installed APU instead you have to tell configure the path to the `apu-config` script. You may set the absolute path and name or the directory to the installed APU. `apu-config` must exist within this directory or the subdirectory `bin`.

**--with-ssl=DIR**

If `mod_ssl` has been enabled configure searches for an installed OpenSSL. You can set the directory path to the SSL/TLS toolkit instead.

**--with-z=DIR**

configure searches automatically for an installed `zlib` library if your source configuration requires one (e.g., when `mod_deflate` is enabled). You can set the directory path to the compression library instead.

Several features of the Apache HTTP Server, including `mod_authn_dbm` and `mod_rewrite`'s DBM `RewriteMap` use simple key/value databases for quick lookups of information. SDBM is included in the APU, so this database is always available. If you would like to use other database types, use the following options to enable them:

**--with-gdbm[=path]**

If no *path* is specified, `configure` will search for the include files and libraries of a GNU DBM installation in the usual search paths. An explicit *path* will cause `configure` to look in *path/lib* and *path/include* for the relevant files. Finally, the *path* may specify specific include and library paths separated by a colon.

**--with-ndbm[=*path*]**

Like `--with-gdbm`, but searches for a New DBM installation.

**--with-berkeley-db[=*path*]**

Like `--with-gdbm`, but searches for a Berkeley DB installation.

**Note**

The DBM options are provided by the APU and passed through to its configuration script. They are useless when using an already installed APU defined by `--with-apr-util`.

You may use more than one DBM implementation together with your HTTP server. The appropriated DBM type will be configured within the runtime configuration at each time.

## Options for support programs

**--enable-static-support**

Build a statically linked version of the support binaries. This means, a stand-alone executable will be built with all the necessary libraries integrated. Otherwise the support binaries are linked dynamically by default.

**--enable-suexec**

Use this option to enable [suexec](#), which allows you to set uid and gid for spawned processes. **Do not use this option unless you understand all the security implications of running a suid binary on your server.** Further options to

configure [suexec](#) are described [below](#).

It is possible to create a statically linked binary of a single support program by using the following options:

**--enable-static-ab**

Build a statically linked version of [ab](#).

**--enable-static-checkgid**

Build a statically linked version of [checkgid](#).

**--enable-static-htdbm**

Build a statically linked version of [htdbm](#).

**--enable-static-htdigest**

Build a statically linked version of [htdigest](#).

**--enable-static-htpasswd**

Build a statically linked version of [htpasswd](#).

**--enable-static-logresolve**

Build a statically linked version of [logresolve](#).

**--enable-static-rotatelogs**

Build a statically linked version of [rotatelogs](#).

### **suexec configuration options**

The following options are used to fine tune the behavior of [suexec](#). See [Configuring and installing suEXEC](#) or further information.

**--with-suexec-bin**

This defines the path to [suexec](#) binary. Default is `--sbindir` (see [Fine tuning of installation directories](#)).

**--with-suexec-caller**

This defines the user allowed to call [suexec](#). It should be the same as the user under which [httpd](#) normally runs.

### **--with-suexec-docroot**

This defines the directory tree under which [suexec](#) access is allowed for executables. Default value is --datadir/htdocs.

### **--with-suexec-gidmin**

Define this as the lowest GID allowed to be a target user for [suexec](#). The default value is 100.

### **--with-suexec-logfile**

This defines the filename of the [suexec](#) logfile. By default the logfile is named suexec\_log and located in --logfiledir.

### **--with-suexec-safepath**

Define the value of the environment variable PATH to be set for processes started by [suexec](#). Default value is /usr/local/bin:/usr/bin:/bin.

### **--with-suexec-userdir**

This defines the subdirectory under the user's directory that contains all executables for which [suexec](#) access is allowed. This setting is necessary when you want to use [suexec](#) together with user-specific directories (as provided by [mod\\_userdir](#)). The default is public\_html.

### **--with-suexec-uidmin**

Define this as the lowest UID allowed to be a target user for [suexec](#). The default value is 100.

### **--with-suexec-umask**

Set umask for processes started by [suexec](#). It defaults to your system settings.



---

There are some useful environment variables to override the choices made by `configure` or to help it to find libraries and programs with nonstandard names or locations.

## **CC**

Define the C compiler command to be used for compilation.

## **CFLAGS**

Set C compiler flags you want to use for compilation.

## **CPP**

Define the C preprocessor command to be used.

## **CPPFLAGS**

Set C/C++ preprocessor flags, e.g. `-Iincludedir` if you have headers in a nonstandard directory *includedir*.

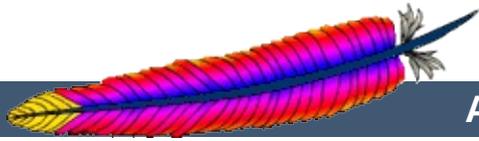
## **LDFLAGS**

Set linker flags, e.g. `-Llibdir` if you have libraries in a nonstandard directory *libdir*.

---

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](#) > [Programs](#)

## dbmmanage - Manage user authentication files in DBM format

dbmmanage is used to create and update the DBM format files used to store usernames and password for basic authentication of HTTP users via [mod\\_auth\\_dbm](#). Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by dbmmanage. This program can only be used when the usernames are stored in a DBM file. To use a flat-file database see [htpasswd](#).

This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in [httpd](#) see the httpd manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

### See also

[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***

The filename of the DBM format file. Usually without the extension `.db`, `.pag`, or `.dir`.

### ***username***

The user for which the operations are performed. The *username* may not contain a colon (`:`).

### ***encpasswd***

This is the already encrypted password to use for the `update` and `add` commands. You may use a hyphen (`-`) if you want to get prompted for the password, but fill in the fields afterwards. Additionally when using the `update` command, a period (`.`) keeps the original password untouched.

### ***group***

A group, which the user is member of. A groupname may not contain a colon (`:`). You may use a hyphen (`-`) if you don't want to assign the user to a group, but fill in the comment field. Additionally when using the `update` command, a period (`.`) keeps the original groups untouched.

### ***comment***

This is the place for your opaque comments about the user, like `realname`, `mailaddress` or such things. The server will ignore this field.

## **Encodings**

### **-d**

crypt encryption (default, except on Win32, Netware)

### **-m**

MD5 encryption (default on Win32, Netware)

### **-s**

SHA1 encryption

**-p**

plaintext (*not recommended*)

## Commands

### add

Adds an entry for *username* to *filename* using the encrypted password *enpasswd*.

```
dbmmanage passwords.dat add rbowen foKntnEF3KSXA
```

### adduser

Asks for a password and then adds an entry for *username* to *filename*.

```
dbmmanage passwords.dat adduser krietz
```

### check

Asks for a password and then checks if *username* is in *filename* and if its password matches the specified one.

```
dbmmanage passwords.dat check rbowen
```

### delete

Deletes the *username* entry from *filename*.

```
dbmmanage passwords.dat delete rbowen
```

### import

Reads *username:password* entries (one per line) from STDIN and adds them to *filename*. The passwords already have to be crypted.

### update

Same as the `adduser` command, except that it makes sure *username* already exists in *filename*.

```
dbmmanage passwords.dat update rbowen
```

### **view**

Just displays the contents of the DBM file. If you specify a *username*, it displays the particular record only.

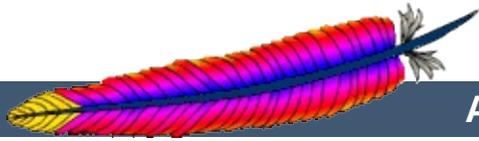
```
dbmmanage passwords.dat view
```



One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, the GNU project's GDBM, and Berkeley DB 2. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by *filename* is the same format that `dbmmanage` expects to see. `dbmmanage` currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

`dbmmanage` has a list of DBM format preferences, defined by the `@AnyDBM: :ISA` array near the beginning of the program. Since we prefer the Berkeley DB 2 file format, the order in which `dbmmanage` will look for system libraries is Berkeley DB 2, then NDBM, then GDBM and then SDBM. The first library found will be the library `dbmmanage` will attempt to use for all DBM file transactions. This ordering is slightly different than the standard `@AnyDBM: :ISA` ordering in Perl, as well as the ordering used by the simple `dbmopen( )` call in Perl, so if you use any other utilities to manage your DBM files, they must also follow this preference ordering. Similar care must be taken if using programs in other languages, like C, to access these files.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.



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



## Apache HTTP Server Version 2.0

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

## htdigest - manage user files for digest authentication

htdigest is used to create and update the flat-files used to store usernames, realm and password for digest authentication of HTTP users. Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htdigest.

This manual page only lists the command line arguments. For details of the directives necessary to configure digest authentication in [httpd](#) see the Apache manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

### See also

[httpd](#)

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



## Synopsis

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



## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is deleted first.

### ***passwdfile***

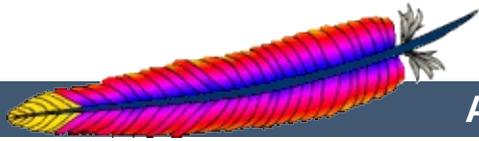
Name of the file to contain the username, realm and password. If -c is given, this file is created if it does not already exist, or deleted and recreated if it does exist.

### ***realm***

The realm name to which the user name belongs.

### ***username***

The user name to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.



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



## Apache HTTP Server Version 2.0

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

## htpasswd - Manage user files for basic authentication

htpasswd is used to create and update the flat-files used to store usernames and password for basic authentication of HTTP users. If htpasswd cannot access a file, such as not being able to write to the output file or not being able to read the file in order to update it, it returns an error status and makes no changes.

Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htpasswd. This program can only manage usernames and passwords stored in a flat-file. It can encrypt and display password information for use in other types of data stores, though. To use a DBM database see [dbmmanage](#).

htpasswd encrypts passwords using either a version of MD5 modified for Apache, or the system's crypt ( ) routine. Files managed by htpasswd may contain both types of passwords; some user records may have MD5-encrypted passwords while others in the same file may have passwords encrypted with crypt ( ).

This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in [httpd](#) see the Apache manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

### See also

[httpd](#)

The scripts in support/SHA1 which come with the distribution.



## Syntax

```
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**

Use batch mode; *i.e.*, get the password from the command line rather than prompting for it. This option should be used with extreme care, since **the password is clearly visible** on the command line.

## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is rewritten and truncated. This option cannot be combined with the `-n` option.

## **-n**

Display the results on standard output rather than updating a file. This is useful for generating password records acceptable to Apache for inclusion in non-text data stores. This option changes the syntax of the command line, since the *passwdfile* argument (usually the first one) is omitted. It cannot be combined with the `-c` option.

## **-m**

Use MD5 encryption for passwords. On Windows, Netware and TPF, this is the default.

## **-d**

Use `crypt ( )` encryption for passwords. The default on all platforms but Windows, Netware and TPF. Though possibly supported by `htpasswd` on all platforms, it is not supported by the [httpd](#) server on Windows, Netware and TPF.

## **-s**

Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (ldif).

## **-p**

Use plaintext passwords. Though `htpasswd` will support

creation on all platforms, the [httpd](#) daemon will only accept plain text passwords on Windows, Netware and TPF.

**-D**

Delete user. If the username exists in the specified `htpasswd` file, it will be deleted.

***passwdfile***

Name of the file to contain the user name and password. If `-c` is given, this file is created if it does not already exist, or rewritten and truncated if it does exist.

***username***

The username to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.

***password***

The plaintext password to be encrypted and stored in the file. Only used with the `-b` flag.



---

htpasswd returns a zero status ("true") if the username and password have been successfully added or updated in the *passwdfile*. htpasswd returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn't match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the [Restrictions section](#)), and 7 if the file is not a valid password file.



## Examples

```
htpasswd /usr/local/etc/apache/.htpasswd-users jsmith
```

Adds or modifies the password for user `jsmith`. The user is prompted for the password. If executed on a Windows system, the password will be encrypted using the modified Apache MD5 algorithm; otherwise, the system's `crypt ( )` routine will be used. If the file does not exist, `htpasswd` will do nothing except return an error.

```
htpasswd -c /home/does/public_html/.htpasswd jane
```

Creates a new file and stores a record in it for user `jane`. The user is prompted for the password. If the file exists and cannot be read, or cannot be written, it is not altered and `htpasswd` will display a message and return an error status.

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

Encrypts the password from the command line (`Pwd4Steve`) using the MD5 algorithm, and stores it in the specified file.



## Security Considerations

Web password files such as those managed by `htpasswd` should *not* be within the Web server's URI space -- that is, they should not be fetchable with a browser.

The use of the `-b` option is discouraged, since when it is used the unencrypted password appears on the command line.



## Restrictions

On the Windows and MPE platforms, passwords encrypted with `htpasswd` are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

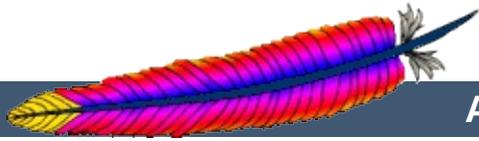
The MD5 algorithm used by `htpasswd` is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Username are limited to 255 bytes and may not include the character `:`.

---

Copyright 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](#) > [Programs](#)

## **logresolve - Resolve IP-addresses to hostnames in Apache log files**

logresolve is a post-processing program to resolve IP-addresses in Apache's access logfiles. To minimize impact on your nameserver, logresolve has its very own internal hash-table cache. This means that each IP number will only be looked up the first time it is found in the log file.

Takes an Apache log file on standard input. The IP addresses must be the first thing on each line and must be separated from the remainder of the line by a space.



## Synopsis

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



## **-s *filename***

Specifies a filename to record statistics.

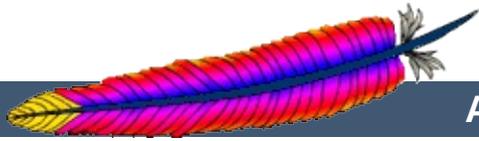
## **-c**

This causes `logresolve` to apply some DNS checks: after finding the hostname from the IP address, it looks up the IP addresses for the hostname and checks that one of these matches the original address.

---

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](#) > [Programs](#)

## rotatelog - Piped logging program to rotate Apache logs

rotatelog is a simple program for use in conjunction with Apache's piped logfile feature. For example:

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

This creates the files `/var/logs/logfile.nnnn` where `nnnn` is the system time at which the log nominally starts (this time will always be a multiple of the rotation time, so you can synchronize cron scripts with it). At the end of each rotation time (here after 24 hours) a new log is started.

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

This configuration will rotate the logfile whenever it reaches a size of 5 megabytes.

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

This configuration will rotate the error logfile whenever it reaches a size of 5 megabytes, and the suffix to the logfile name will be created of the form `errorlog.YYYY-mm-dd-HH_MM_SS`.



## Synopsis

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



## **-1 (2.0.51 and later)**

Causes the use of local time rather than GMT as the base for the interval. Note that using `-1` in an environment which changes the GMT offset (such as for BST or DST) can lead to unpredictable results!

## ***logfile***

The path plus basename of the logfile. If *logfile* includes any '%' characters, it is treated as a format string for `strftime(3)`. Otherwise, the suffix `.nnnnnnnnnn` is automatically added and is the time in seconds. Both formats compute the start time from the beginning of the current period.

## ***rotationtime***

The time between log file rotations in seconds.

## ***offset***

The number of minutes offset from UTC. If omitted, zero is assumed and UTC is used. For example, to use local time in the zone UTC -5 hours, specify a value of `-300` for this argument.

## ***filesizeM***

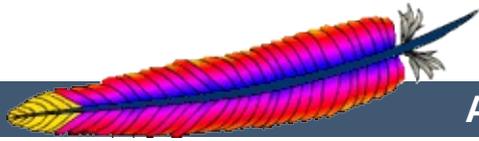
The maximum file size in megabytes followed by the letter M to specify size rather than time. Use this parameter in place of both *rotationtime* and *offset*.



The following logfile format string substitutions should be supported by all `strftime(3)` implementations, see the `strftime(3)` man page for library-specific extensions.

%A	full weekday name (localized)
%a	3-character weekday name (localized)
%B	full month name (localized)
%b	3-character month name (localized)
%c	date and time (localized)
%d	2-digit day of month
%H	2-digit hour (24 hour clock)
%I	2-digit hour (12 hour clock)
%j	3-digit day of year
%M	2-digit minute
%m	2-digit month
%p	am/pm of 12 hour clock (localized)
%S	2-digit second
%U	2-digit week of year (Sunday first day of week)
%W	2-digit week of year (Monday first day of week)
%w	1-digit weekday (Sunday first day of week)
%X	time (localized)
%x	date (localized)
%Y	4-digit year
%y	2-digit year
%Z	time zone name
%%	literal `%'

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



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



## Apache HTTP Server Version 2.0

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

## Other Programs

The following programs are simple support programs included with the Apache HTTP Server which do not have their own manual pages. They are not installed automatically. You can find them after the configuration process in the `support/` directory.



This perl script is designed to be run at a frequent interval by something like cron. It connects to the server and downloads the status information. It reformats the information to a single line and logs it to a file. Adjust the variables at the top of the script to specify the location of the resulting logfile.



This perl script will take a combined Web server access log file and break its contents into separate files. It assumes that the first field of each line is the virtual host identity (put there by "%v"), and that the logfiles should be named that + ".log" in the current directory.

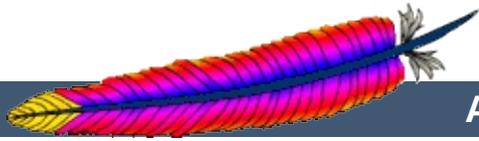
The combined log file is read from stdin. Records read will be appended to any existing log files.

```
split-logfile < access.log
```

---

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](#)

# 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.



## Introduction

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.

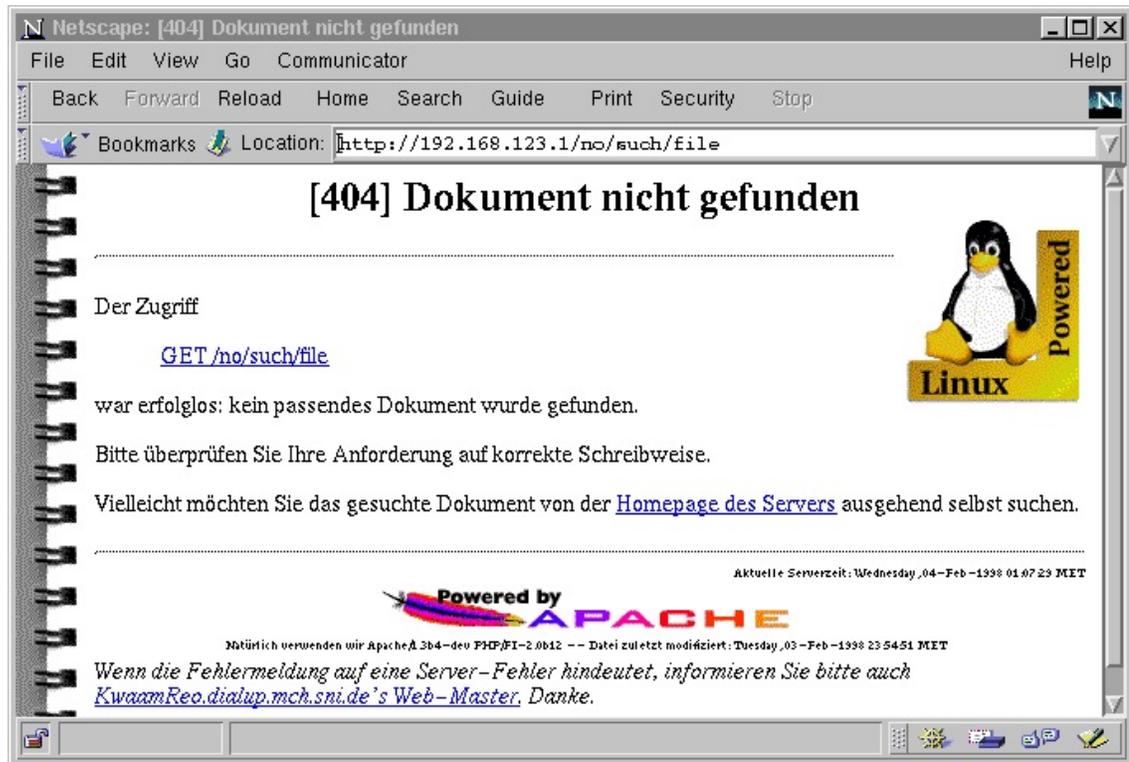
Error documents can be shared across different servers, or even hosts, because all varying information is inserted at the time the error document is returned on behalf of a failed request.

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 /errordocs/404  
# "500 Internal Server Error",  
ErrorDocument 500 /errordocs/500
```

The directory for the error messages (here: `/usr/local/apache/errordocs/`) 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 `errordocs/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 `errordocs/403.shtml.lang` is created and filled with the error text in that language ([see below](#)).
- One fallback document called `errordocs/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>

```

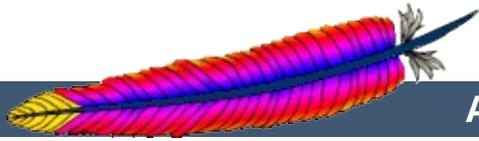
---

If you have tips to contribute, send mail to [martin@apache.org](mailto: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

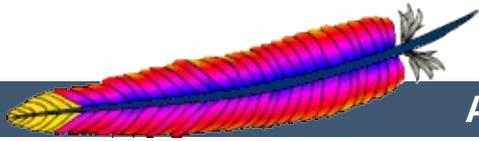
write half of the connection (what shutdown is supposed to do), and continue reading on the socket until it is either closed by the client (signifying it has finally read the response) or a timeout occurs. That is what the kernel is supposed to do if `SO_LINGER` is set. Unfortunately, `SO_LINGER` has 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 with 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:

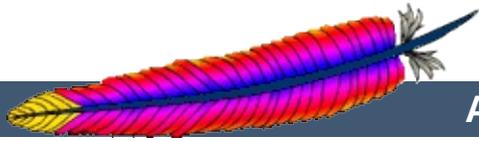
```
BrowserMatch "MSIE 4\.0" force-no-vary
```

(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.



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`



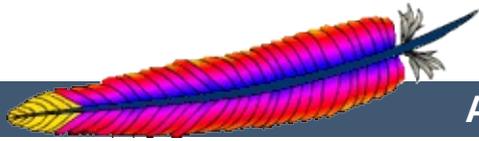
If you have details on another operating system, please submit it through our [Bug Report Page](#).

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.



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



## Apache HTTP Server Version 2.0

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

## Relevant Standards

This page documents all the relevant standards that the Apache HTTP Server follows, along with brief descriptions.

In addition to the information listed below, the following resources should be consulted:

- <http://purl.org/NET/http-errata> - HTTP/1.1 Specification Errata
- <http://www.rfc-editor.org/errata.html> - RFC Errata
- <http://ftp.ics.uci.edu/pub/ietf/http/#RFC> - A pre-compiled list of HTTP related RFCs

### Notice

This document is not yet complete.



Regardless of what modules are compiled and used, Apache as a basic web server complies with the following IETF recommendations:

**[RFC 1945](#) (Informational)**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. This documents HTTP/1.0.

**[RFC 2616](#) (Standards Track)**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This documents HTTP/1.1.

**[RFC 2396](#) (Standards Track)**

A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.



---

Regarding the Hypertext Markup Language, Apache complies with the following IETF and W3C recommendations:

#### **[RFC 2854 \(Informational\)](#)**

This document summarizes the history of HTML development, and defines the "text/html" MIME type by pointing to the relevant W3C recommendations.

#### **[HTML 4.01 Specification \(Errata\)](#)**

This specification defines the HyperText Markup Language (HTML), the publishing language of the World Wide Web. This specification defines HTML 4.01, which is a subversion of HTML 4.

#### **[HTML 3.2 Reference Specification](#)**

The HyperText Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents.

#### **[XHTML 1.1 - Module-based XHTML \(Errata\)](#)**

This Recommendation defines a new XHTML document type that is based upon the module framework and modules defined in Modularization of XHTML.

#### **[XHTML 1.0 The Extensible HyperText Markup Language \(Second Edition\) \(Errata\)](#)**

This specification defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4.



Concerning the different methods of authentication, Apache follows the following IETF recommendations:

**[RFC 2617](#) (Draft standard)**

"HTTP/1.0", includes the specification for a Basic Access Authentication scheme.



The following links document ISO and other language and country code information:

### [ISO 639-2](#)

ISO 639 provides two sets of language codes, one as a two-letter code set (639-1) and another as a three-letter code set (this part of ISO 639) for the representation of names of languages.

### [ISO 3166-1](#)

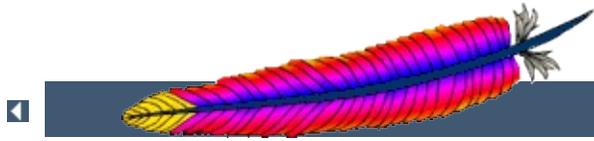
These pages document the country names (official short names in English) in alphabetical order as given in ISO 3166-1 and the corresponding ISO 3166-1-alpha-2 code elements.

### [BCP 47 \(Best Current Practice\), RFC 3066](#)

This document describes a language tag for use in cases where it is desired to indicate the language used in an information object, how to register values for use in this language tag, and a construct for matching such language tags.

### [RFC 3282 \(Standards Track\)](#)

This document defines a "Content-language:" header, for use in cases where one desires to indicate the language of something that has RFC 822-like headers, like MIME body parts or Web documents, and an "Accept-Language:" header for use in cases where one wishes to indicate one's preferences with regard to language.



| | [FAQ](#) | |



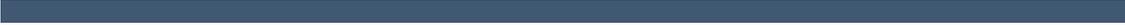
## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache

Apache





---

Apache

**MPM**

"MPM" Apache MPM

**Base**

"Base"

**Extension**

"Extension"

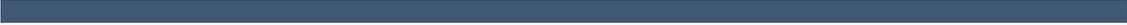
**Experimental**

"Experimental" Apache

**External**

"External" Apache ("")





---

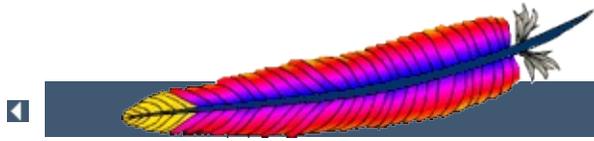
## LoadModule



## Apache 2

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

| | [FAQ](#) | |



| | [FAQ](#) | |



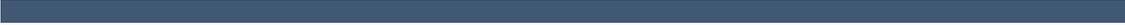
## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)



Apache





"..."

**URL**

http://www.example.com/path/to/file.html  
( Uniform Resource Loc:

**URL-path**

/path/to/file.html url

**file-path**

/usr/local/apache/htdocs/path/to/file.html  
file-path [ServerRoot](#)

**directory-path**

/usr/local/apache/htdocs/path/to/

**filename**

file.html

**regex**

Perl regex

**extension**

filename Apache  
:) filename file.html.en  
Apache extension

**MIME-type**

text/html

**env-variable**

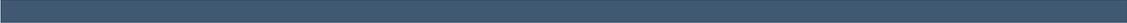
Apache



---

( Apache





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

<VirtualHost>

<Directory>, <Location>, <Files:

Location, Files

**.htaccess**  
.htaccess

httpd.conf .htaccess (: Boolean OR  
<VirtualHost> <Direct



---

.htaccess

AllowOverride

()



---

Apache

**Core**

"Core" Apache

**MPM**

"MPM"

**Base**

"Base"

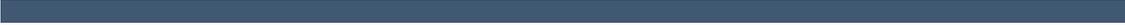
**Extension**

"Extension" Apache

**Experimental**

"Experimental" Apache



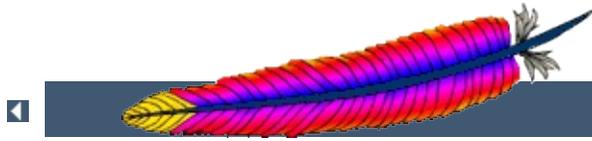


## Apache 2

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache

This translation may be out of date. Check the English version for recent changes.

 Apache HTTP

 Core



## AcceptPathInfo

```
:\
:\ AcceptPathInfo On|Off|Default
:\ AcceptPathInfo Default
:\ , , , .htaccess
:\ FileInfo
:\ Core
:\ core
:\ Apache 2.0.30
```

```
(
    /test/ here.html
    /test/nothere.html/more /more P/
```

**AcceptPathInfo** :

**off**

**On**

**Default**

PATH\_INFO

```
AcceptPathInfo PATH_INFO
PATH_INFO
```

```
<Files "mypaths.shtml">
  Options +Includes
  SetOutputFilter INCLUDES
  AcceptPathInfo On
```

</Files>



```
AccessFileName filename [filename] ...  
AccessFileName .htaccess  
,  
Core  
core
```

```
AccessFileName .acl
```

```
AccessFileName .acl, /usr/local/web/.acl
```

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

- [AllowOverride](#)
- [.htaccess](#)



## ADDDEFAULTCHARSET

```
⋮  
⋮ AddDefaultCharset On|Off|charset  
⋮ AddDefaultCharset Off  
⋮ , , , .htaccess  
⋮ FileInfo  
⋮ Core  
⋮ core
```

## HTTP

```
1 AddDefaultCharset On  
   charset :
```

```
AddDefaultCharset utf-8
```



## AddOutputFilterByType

```
: MIME-type
: AddOutputFilterByType filter[;filter...]
  MIME-type [MIME-type] ...
: , , , .htaccess
: FileInfo
: Core
: core
: Apache 2.0.33
```

### MIME-type

```
mod_deflate DEFLATE text/html te
()
```

```
AddOutputFilterByType DEFLATE text/html text/plain
```

```
text/html INCLUDES DEFLAT
```

```
<Location /cgi-bin/>
  Options Includes
  AddOutputFilterByType INCLUDES;DEFLATE text/html
</Location>
```

```
:
AddOutputFilterByType
DefaultType DefaultType
```

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



```
URL
AllowEncodedSlashes On|Off
AllowEncodedSlashes Off
,
Core
core
Apache 2.0.46
```

AllowEncodedSlashes ( / %:
URL 404 (Not found)

AllowEncodedSlashes On PATH\_INFO

Turning AllowEncodedSlashes On is mostly useful when used
in conjunction with PATH\_INFO.

```
%2F () %5C URL
```

- [AcceptPathInfo](#)



```

: .htaccess
: AllowOverride All|None|directive-type
  [directive-type] ...
: AllowOverride All
:
: Core
: core

```

( [AccessFileName](#) ) .htaccess

```

<Directory>
  AllowOverride <Directory> <Location>
  <DirectoryMatch> <Files>

```

None [.htaccess](#)

All .htaccess

*directive-type*

### AuthConfig

```

(
  AuthDBMGroupFile, AuthDBMUserFile,
  AuthGroupFile, AuthName, AuthType, AuthUserFile,
  Require )

```

### FileInfo

```

(
  DefaultLanguage, DefaultLanguagePriority,
  LanguagePriority, SetHandler, SetInputFilter,
  SetOutputFilter, mod\_mime Add* Remove*
)

```

### Indexes

```

(AddIcon, AddIconByType, DefaultIcon, DirectoryIndex,

```

FancyIndexing, HeaderName, IndexIgnore,  
IndexOptions, ReadmeName )

## Limit

(

Allow

## Options

:

AllowOverride AuthConfig Indexes

AuthConfig Indexes

- AccessFileName
- 
- .htaccess



AuthName

```
[: HTTP (: realm)
[: AuthName auth-domain
[: , .htaccess
[: AuthConfig
[: Core
[: core
```

(: realm)

Require                    AuthUserFile   AuthGroupFile

:

```
AuthName "Top Secret"
```

AuthName

- 



```
AuthType Basic|Digest  
, .htaccess  
AuthConfig  
Core  
core
```

AuthUserFile AuthGroupFile

- 



## Configuration

```
[:] CGI
[:] CGIMapExtension cgi-path .extension
[:] None
[:] , .htaccess
[:] FileInfo
[:] Core
[:] core
[:] NetWare
```

Apache CGI

.foo .foo CGI FOO



Content-Digest

- [Content -MD5 HTTP](#)
- [ContentDigest On|Off](#)
- [ContentDigest Off](#)
- [, , , .htaccess](#)
- [Options](#)
- [Core](#)
- [core](#)

RFC1864 RFC2068 Conte

MD5 ( )

Content -MD5

:

Content-MD5: AuLb7Dp1rqtRtxz2m9kRpA==

()

Content -MD5 [core](#)



## DefaultType

```
: MIME
: DefaultType MIME-type
: DefaultType text/plain
: , , , .htaccess
: FileInfo
: Core
: core
```

### MIME

```
DefaultType image/gif
```

.gif

GIF

ForceType

MIME



## Directory

```

:
: <Directory directory-path> ... </Directory>
: ,
: Core
: core

```

```

path Unix ? 1
    /home/user/public_html <Directory
/* /public_html> <Directory
/home/* /public_html> :
```

```

<Directory /usr/local/httpd/htdocs>
  Options Indexes FollowSymLinks
</Directory>

```

```

directory-path : Apache <Directory>

```

```

~ :
```

```

<Directory ~ "^/www/.*/[0-9]{3}">

```

```

  /www/ 3
() <Directory> ()

```

```

<Directory />
  AllowOverride None
</Directory>

<Directory /home/>
  AllowOverride FileInfo
</Directory>

```

---

/home/web/dir/doc.html :

- AllowOverride None (.htaccess)
- AllowOverride FileInfo (/home)
- /home/.htaccess, /home/web/.htaccess, /home/web/.htaccess FileInfo

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

```
<Directory> .htaccess  
/home/abc/public_html/abc <Directory>
```

**Apache <Directory /> Allow from All**  
**URL Apache**

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

```
httpd.conf <Directory>  
<LimitExcept>
```

•



## DirectoryMatch

```

:
: <DirectoryMatch regex> ... </DirectoryMatch>
: ,
: Core
: core

```

```
<Directory> <DirectoryMatch> </Dir
```

```
<DirectoryMatch "^/www/.*/[0-9]{3}">
```

/www/ 3

- <Directory> <Directory>
- 



## DocumentRoot

```
DocumentRoot directory-path
DocumentRoot /usr/local/apache/htdocs
,
Core
core
```

httpd

Alias

```
DocumentRoot /usr/web
```

```
http://www.my.host.com/index.html
/usr/web/index.html
```

**DocumentRoot**

- URL



```
EnableMMAP On|Off
EnableMMAP On
, , , .htaccess
FileInfo
Core
core
```

httpd

- httpd
- NFS DocumentRoot httpd

:

```
EnableMMAP Off
```

NFS

:

```
<Directory "/path-to-nfs-files">
  EnableMMAP Off
</Directory>
```



```
sendfile
EnableSendfile On|Off
EnableSendfile On
, , , .htaccess
FileInfo
Core
core
2.0.44
```

httpd se

sendfile read send

- sendfile
- Linux sendfile IPv6 TCP-checksum
- DocumentRoot ( NFS SMB)

:

```
EnableSendfile Off
```

NFS SMB :

```
<Directory "/path-to-nfs-files">
  EnableSendfile Off
</Directory>
```



```

:
: ErrorDocument error-code document
: , , , .htaccess
: FileInfo
: Core
: core
: Apache 2.0

```

Apache

1. Apache

2.

3.

*URL-path*

4. *URL*

2 4

**ErrorDocument**

Apache

URL URL (/)

URI

:

```

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"

```

default Apache

**ErrorDocument**

Apache

```

ErrorDocument 404 /cgi-bin/bad_urls.pl

<Directory /web/docs>
  ErrorDocument 404 default
</Directory>

```

URL ( http ) ErrorDocument  
Apache  
ErrorDocument 401 URL 401  
**ErrorDocument 401**

Microsoft Internet Explorer (MSIE)  
MSIE Microsoft  
2.0

• |



## ErrorLog

```
:  
: ErrorLog file-path|syslog[:facility]  
: ErrorLog logs/error_log (Unix) ErrorLog  
: logs/error.log (Windows and OS/2)  
: ,  
: Core  
: core
```

### ErrorLog

```
ErrorLog /var/log/httpd/error_log
```

#### *file-path* ()

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

```
syslog syslogd(8)  
syslog:facility syslog(1)
```

```
ErrorLog syslog:user
```

:

```
Unix
```

- [LogLevel](#)
- [Apache](#)



## FileETag

- ETag HTTP
- FileETag *component* ...
- FileETag INode MTime Size
- , , , .htaccess
- FileInfo
- Core
- core

FileETag E

ETag inode, (mtime)

### INode

inode

### MTime

### Size

### All

FileETag INode MTime Size

### None

ETag

INode, MTime, Size + -

( FileETag INode MTime Size )



```
⋮
⋮ <Files filename> ... </Files>
⋮ , , , .htaccess
⋮ All
⋮ Core
⋮ core
```

```
<Files>
  </Files>                                ()
.htaccess                                <Location>
<Directory>
filename                                ?
```

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

```
<Directory> <Location> <Files> .
```

•



## FILESMATCH

```
⋮  
⋮ <FilesMatch regex> ... </FilesMatch>  
⋮ , , , .htaccess  
⋮ All  
⋮ Core  
⋮ core
```

<FilesMatch>

<Files>

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

•





HostnameLookups

```
: IP DNS
: HostnameLookups On|Off|Double
: HostnameLookups Off
: ,
: Core
: core
```

DNS

IP

mod\_access

Double 2  
REMOTE\_HOST

2

bin



## Identity Check

```
⋮ RFC1413
⋮ IdentityCheck On|Off
⋮ IdentityCheck Off
⋮ ,
⋮ Core
⋮ core
```

identd



```
⋮
⋮ <IfDefine [!]parameter-name> ... </IfDefine>
⋮ , , , .htaccess
⋮ All
⋮ Core
⋮ core
```

```
<IfDefine test>...</IfDefine>
  test                                test

<IfDefine>                            test  :
  • parameter-name
  • !parameter-name

  parameter-name

parameter-name                        httpd          -Dpa

<IfDefine>
```

```
httpd -DReverseProxy ...

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



## MODULES

```
⋮  
⋮ <IfModule [!]module-name> ... </IfModule>  
⋮ , , , .htaccess  
⋮ All  
⋮ Core  
⋮ core
```

```
<IfModule test>...</IfModule>  
test test
```

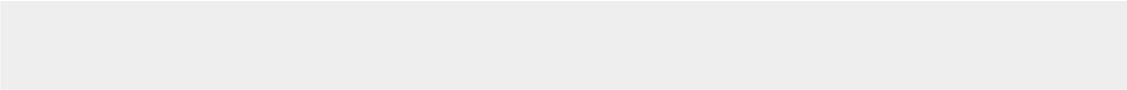
```
<IfModule> test
```

- *module name*
- *!module name*

```
module name Apache (  
module name
```

```
module name m  
STANDARD20_MODULE_STUFF
```

```
<IfModule>
```



```
Include file-path|directory-path
,
Core
core
2.0.41
```

( fnmatch)

httpd

ServerRoot

:

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

ServerRoot :

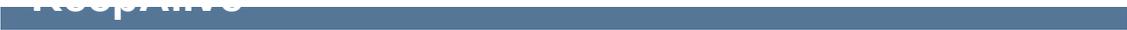
```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

apachectl configtest :

```
root@host# apachectl configtest
Processing config file: /usr/local/apache2/conf/ssl.conf
Processing config file:
/usr/local/apache2/conf/vhosts/vhost1.conf
Processing config file:
/usr/local/apache2/conf/vhosts/vhost2.conf
Syntax OK
```

- [apachectl](#)





```
HTTP
KeepAlive On|Off
KeepAlive On
,
Core
core
```

HTTP/1.0 Keep-Alive HTTP/1.1 TCP  
HTML 50%

HTTP/1.0 Keep-Alive HTTP/1.1 Keep-Alive  
Keep-Alive CGI SSI  
HTTP/1.1

- MaxKeepAliveRequests



## KeepAliveTimeout

```
KeepAliveTimeout seconds
KeepAliveTimeout 15
,
Core
core
```

Apache

KeepAliveTimeout



```
: HTTP
: <Limit method [method] ... > ... </Limit>
: , , , .htaccess
: All
: Core
: core
```

<Limit> HTTP  
DELETE :

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

: GET, POST, PUT  
PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK.  
GET HEAD TRACE

```
<Limit> <LimitExcept>
```



## LimitExcept

```
: HTTP
: <LimitExcept method [method] ... > ...
:   </LimitExcept>
: , , , .htaccess
: All
: Core
: core
```

```
<LimitExcept> </LimitExcept> HTTP
  <Limit>
:
```

```
<LimitExcept POST GET>
  Require valid-user
</LimitExcept>
```



## LIMITINTERNALRECURSION

```
LimitInternalRecursion number [number]  
LimitInternalRecursion 10  
,  
Core  
core  
Apache 2.0.47
```

Action mod\_dir Action DirectoryIndex

LimitInternalRecursion

```
LimitInternalRecursion 5
```



LimitRequestBody

- ⋮ HTTP
- ⋮ LimitRequestBody bytes
- ⋮ LimitRequestBody 0
- ⋮ , , , .htaccess
- ⋮ All
- ⋮ Core
- ⋮ core

bytes 0 () 2147483647 (2GB)

LimitRequestBody (

100K

LimitRequestBody 102400



## LimitRequestFields

```
: HTTP
: LimitRequestFields number
: LimitRequestFields 100
:
: Core
: core
```

*number* 0 () 32767  
DEFAULT\_LIMIT\_REQUEST\_FIELDS ( 100 )

LimitRequestBody HTTP  
HTTP

:

LimitRequestFields 50



## LimitRequestFieldSize

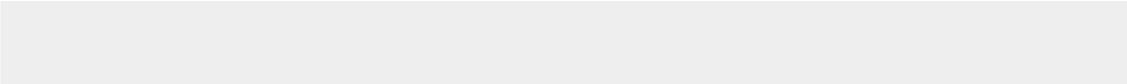
```
⋮ HTTP
⋮ LimitRequestFieldSize bytes
⋮ LimitRequestFieldSize 8190
⋮
⋮ Core
⋮ core
```

HTTP by  
DEFAULT\_LIMIT\_REQUEST\_FIELD\_SIZE ( 8192 )

### LimitRequestFieldSize

:

```
LimitRequestFieldSize 4094
```



LimitRequestLine

- [: HTTP](#)
- [: LimitRequestLine bytes](#)
- [: LimitRequestLine 8190](#)
- [:](#)
- [: Core](#)
- [: core](#)

HTTP *bytes 0*  
 8190 )

LimitRequestLine  
 LimitRequestLine URI

:

LimitRequestLine 4094



LimitXMLRequestBody

- XML
- LimitXMLRequestBody *bytes*
- LimitXMLRequestBody 1000000
- , , , .htaccess
- All
- Core
- core

XML () 0

:

LimitXMLRequestBody 0



```

: URL
: <Location URL-path|URL> ... </Location>
: ,
: Core
: core

```

```

<Location> URL </Location>
<Files>
<Location>

```

```

<Location>
<Location>
  <Location />

```

```

() URL /path/ http
scheme://servername/path
URL ? *
~ :
```

```

<Location ~ "/(extra|special)/data">

```

```

URL /extra/data /special/data
<LocationMatch> <Location>

```

```

<Location> SetHandler

```

```

<Location /status>

```

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

```
/()
URL
  <LocationMatch>                                <Location>
    <LocationMatch ^/abc>    /abc URL
URL                                ()                <Location>
<Location> proxy
  /abc//def
```

•



## LocationMatch

```
: URL  
: <LocationMatch regex> ... </LocationMatch>  
: ,  
: Core  
: core
```

<LocationMatch>                      <Location>                      URL

```
<LocationMatch "/(extra|special)/data">
```

URL    /extra/data    /special/data

- 



## LogLevel

```
: ErrorLog
: LogLevel level
: LogLevel warn
: ,
: Core
: core
```

```
LogLevel ( ErrorLog )
```

emerg	-	Child cannot open lock file. Exiting ()
alert		getpwuid: couldn't determine user name from uid (getpwuid: UID )
crit		socket: Failed to get a socket, exiting child (socket: )
error		Premature end of script headers ()
warn		child process 1234 did not exit, sending another SIGHUP ( 1234 SIGHUP )
notice		httpd: caught SIGBUS, attempting to dump core in ... (httpd: SIGBUS ... )
info		"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..." ( StartServers Min/MaxSpareServers )
debug		"Opening config file ..." (...)

crit

:



LogLevel notice

notice



## maxKeepAliveRequests

```
:  
: MaxKeepAliveRequests number  
: MaxKeepAliveRequests 100  
: ,  
: Core  
: core
```

MaxKeepAliveRequests

KeepAlive

:

```
MaxKeepAliveRequests 500
```



## MaxRanges

- [:] Number of ranges allowed before returning the complete resource
- [:] MaxRanges default | unlimited | none | *number-of-ranges*
- [:] MaxRanges 200
- [:] , ,
- [:] Core
- [:] core
- [:] Available in Apache HTTP Server 2.0.65 and later

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



## NAMEVIRTUALHOST

```
IP  
NameVirtualHost addr[:port]  
Core  
core
```

### NameVirtualHost

*addr* IP

```
NameVirtualHost 111.22.33.44
```

NameVirtualHost IP

```
_default_ NameVirtualHost IP  
NameVirtualHost VirtualHost )
```

```
NameVirtualHost 111.22.33.44:8080
```

### IPV6 :

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

\*

```
NameVirtualHost *
```

**<VirtualHost>**

**<VirtualHost> NameVirtualHost**

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



## Options

```
Options [+|-]option [[+|-]option] ...  
Options All  
,, ,.htaccess  
Options  
Core  
core
```

### Options

*option* None 1

### All

MultiViews

### ExecCGI

mod\_cgi CGI

### FollowSymLinks

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

### Includes

mod\_include SSI

### IncludesNOEXEC

SSI #exec #exec CGI  
virtual ScriptAlias CGI

### Indexes

URL Director

mod\_autoindex

**MultiViews**

mod\_negotiation "MultiViews"

**SymLinksIfOwnerMatch**

ID

<Location>

Options

+

+ - :

```
<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
  Options Includes
</Directory>
```

/web/docs/spec

Includes

- :

```
<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
  Options +Includes -Indexes
</Directory>
```

/web/docs/spec

FollowSymLinks

Includes

-IncludesNOEXEC

-Includes

SSI

All



## Require

```
:  
: Require entity-name [entity-name] ...  
: , .htaccess  
: AuthConfig  
: Core  
: core
```

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

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

**Require valid-user**

**Require AuthName AuthType AuthUserFile AuthGroupFile ()**

```
AuthType Basic  
AuthName "Restricted Directory"  
AuthUserFile /web/users  
AuthGroupFile /web/groups  
Require group admin
```

- [Satisfy](#)
- [mod\\_access](#)



```
⋮ Apache CPU
⋮ RLimitCPU seconds|max [seconds|max]
⋮
⋮ , , , .htaccess
⋮ All
⋮ Core
⋮ core
```

Apache  
Apache fork

Apache fork

CPU

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



```
⋮ Apache
⋮ RLimitMEM bytes|max [bytes|max]
⋮
⋮ , , , .htaccess
⋮ All
⋮ Core
⋮ core
```

Apache  
Apache fork

Apache fork

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



```
⋮ Apache
⋮ RLimitNPROC number | max [number | max]
⋮
⋮ , , , .htaccess
⋮ All
⋮ Core
⋮ core
```

Apache  
Apache fork

Apache fork

CGI ID

- [RLimitMEM](#)
- [RLimitCPU](#)



```
:  
: Satisfy Any|All  
: Satisfy All  
: , .htaccess  
: AuthConfig  
: Core  
: core  
: 2.0.51 <Limit>  
    <LimitExcept>
```

Allow Require  
Any

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

2.0.51                    <Limit>                    <LimitExcept>

- Allow
- Require



```

: CGI
: ScriptInterpreterSource Registry|Registry-
  Strict|Script
: ScriptInterpreterSource Script
: , , , .htaccess
: FileInfo
: Core
: core
: Win32      Registry-Strict Apache 2.0

```

```

Apache CGI
)      Win32

```

```

#!C:/Perl/bin/perl.exe

```

```

perl  PATH :

```

```

#!perl

```

```

ScriptInterpreterSource Registry      (
Windows      HKEY_CLASSES_ROOT
Shell\ExecCGI\Command      Shell\Open\C
Apache      Script

```

```

ScriptInterpreterSource Registry ScriptAlias
Apache
      Microsoft Internet Explorer

```

```

Apache 2.0  Registry-Strict  Registry
Shell\ExecCGI\Command      ExecCGI Windows

```



```
ServerAdmin email-address
Core
core
```

ServerAdmin

ServerAdmin www-admin@foo.example.com



```
ServerAlias hostname [hostname] ...  
Core  
core
```

### ServerAlias

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

- [Apache](#)



## ServerName

```
ServerName fully-qualified-domain-name[:port]
Core
core
2.0 1.3 Port
```

ServerName  
simple.example.com DNS www.example.com

```
ServerName www.example.com:80
```

ServerName IP  
ServerName

<VirtualHost> ServerName

URL ( mod\_dir )

- [DNS Apache](#)
- [Apache](#)
- [UseCanonicalName](#)
- [NameVirtualHost](#)
- [ServerAlias](#)



## ServerPath

- URL
- ServerPath *URL-path*
- 
- Core
- core

ServerPath

URL

- [Apache](#)



## ServerRoot

```
ServerRoot directory-path
ServerRoot /usr/local/apache
Core
core
```

## ServerRoot

```
ServerRoot /home/httpd
```

- [httpd -d](#)
- **ServerRoot**



## ServerSignature

```
:  
: ServerSignature On|Off|EMail  
: ServerSignature Off  
: , , , .htaccess  
: All  
: Core  
: core
```

ServerSignature (   
mod\_info )

Off (Apache-1.2)  
ServerName EMail ServerAdmin "mailto:"

### 2.0.44 ServerSignature

- ServerTokens



```
ServerTokens
ServerTokens Full
ServerTokens Major
ServerTokens Minor
ServerTokens Min[imal]
ServerTokens OS
ServerTokens Full ()
```

Server OS

### **ServerTokens Prod[uctOnly]**

(): 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: Apache/2.0.41

### **ServerTokens OS**

(): Server: Apache/2.0.41 (Unix)

### **ServerTokens Full ()**

(): Server: Apache/2.0.41 (Unix) PHP/4.2.2  
MyMod/1.2

2.0.44 [ServerSignature](#)

- [ServerSignature](#)



## SetHandler

```
SetHandler handler-name|None  
[, , , .htaccess  
FileInfo  
Core  
core  
Apache 2.0 core
```

`.htaccess`    <Directory>                    <Location>  
*name*

```
SetHandler imap-file
```

:URL `http://servername/status`

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

None

SetHandler

- AddHandler



```
POST
SetInputFilter filter[;filter...]
,,, .htaccess
FileInfo
Core
core
```

SetInputFilter

POST

•



## SetOutputFilter

```
SetOutputFilter filter[;filter...]  
    , , , .htaccess  
    FileInfo  
    Core  
    core
```

### SetOutputFilter

/www/data/

SSI

```
<Directory /www/data/>  
    SetOutputFilter INCLUDES  
</Directory>
```

- 



## TimeOut

```
TimeOut seconds
TimeOut 300
Core
core
```

TimeOut :

1. GET
2. POST PUT TCP
3. TCP ACK

Apache 1.2 1200



## TRACE\_MODULE

- [\[:\]](#) Determines the behaviour on TRACE requests
- [\[:\]](#) TraceEnable [*on|off|extended*]
- [\[:\]](#) TraceEnable on
- [\[:\]](#)
- [\[:\]](#) Core
- [\[:\]](#) core
- [\[:\]](#) Available in Apache 1.3.34, 2.0.55 and later

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



```

:
: UseCanonicalName On|Off|Dns
: UseCanonicalName On
: ,
: Core
: core

```

Apache URL URL Us  
 ) Apache ServerName Port  
 SERVER\_NAME SERVER\_PORT

UseCanonicalName Off Apache URL  
 CGI SERVER\_NAME SERVER

www URL  
 http://www.domain.com/splat/  
 1 www.domain.com -- FAQ  
 UseCanonicalName Off Apache  
 http://www/splat/

UseCanonicalName DNS Host:  
 IP DNS URL

```

CGI SERVER_NAME
URL

```

- ServerName
- Listen



```

: IP
: <VirtualHost addr[:port] [addr[:port]] ...>
  ... </VirtualHost>
:
: Core
: core

```

<VirtualHost> </VirtualHost>

<VirtualHost> *Addr:*

- IP
- IP
- NameVirtualHost \* IP
- IP IP

```

<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

IPv6 :

```

<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>

```

IP  
alias )

```
:port
)
```

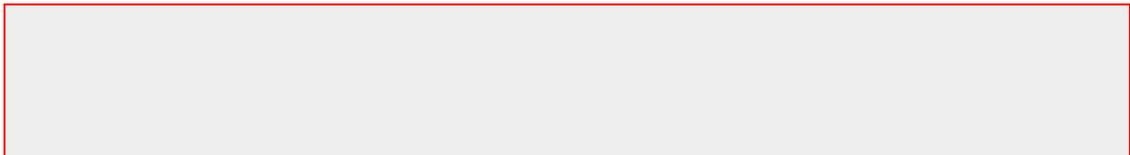
```
:
```

```
<VirtualHost> Apache Listen IP Lis
Apache listen
```

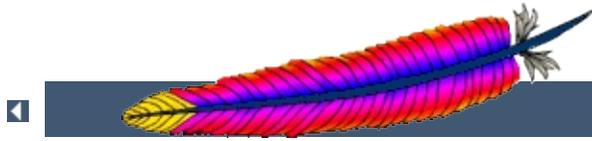
```
IP _default_
VirtualHost (
_default_ )
```

```
:port
)
```

```
:port
)
```



- [Apache](#)
- [DNS Apache](#)
- [Apache](#)
-



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache MPM

This translation may be out of date. Check the English version for recent changes.

 (MPM)

 MPM



## AcceptMutex

```
:| accept Apache
:| AcceptMutex default|method
:| AcceptMutex default
:|
:| MPM
:| leader, perchild, prefork, threadpool, worker
```

AcceptMutex accept

Default

**flock**  
LockFile flock(2)

**fcntl**  
LockFile fcntl(2)

**posixsem**  
POSIX

**pthread**  
POSIX Threads (PThreads) POSIX

**sysvsem**  
SysV

LogLev





## CoreDumpDirectory

```
[: Apache
[: CoreDumpDirectory directory
[:
[:
[: MPM
[: beos, leader, mpm_winnt, perchild, prefork,
threadpool, worker
```

Apache

<b>Linux</b>	
Apache root	Linux
2.4	CoreDumpDirec



## EnableExceptionHook

```
:  
: EnableExceptionHook On|Off  
: EnableExceptionHook Off  
:  
: MPM  
: leader, perchild, prefork, threadpool, worker  
: 2.0.49
```

--enable-exception-hook configure

mod\_whatkilledus n

Trawick [EnableExceptionHook site](#)



## Group

```
:  
: Group unix-group  
: Group #-1  
:  
: MPM  
: beos, leader, mpmt_os2, perchild, prefork,  
  threadpool, worker  
: Apache 2.0
```

### Group

:

#

```
Group www-group
```

nobod

```
Group ( User)
```

: <VirtualHost> Apache 2.0

### SuexecUserGroup

```
Group beos mpmt_os2 MPM
```



```
listen IP
Listen [IP-address:]portnumber
MPM
beos, leader, mpm_netware, mpm_winnt, mpmt_os2,
perchild, prefork, threadpool, worker
Apache 2.0
```

Listen Apache

IP listen

Apache

Listen

listen

Listen

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
```

```
IP Listen 'Address already in use'
```

- [DNS](#)
- [Apache](#)



## ListenBacklog

```
:  
: ListenBacklog backlog  
: ListenBacklog 511  
:  
: MPM  
: beos, leader, mpm_netware, mpm_winnt, mpmt_os2,  
  perchild, prefork, threadpool, worker
```

OS

OS OS



```
⋮  
⋮ LockFile filename  
⋮ LockFile logs/accept.lock  
⋮  
⋮ MPM  
⋮ leader, perchild, prefork, threadpool, worker
```

AcceptMutex fcntl flock  
logs NFS

```
/var/tmp
```

- AcceptMutex



## MAXCLIENTS

```
MaxClients number
MPM beos, leader, prefork, threadpool, worker
```

### MaxClients

```
( prefork ) MaxClients
ServerLimit
```

```
( beos worker ) MaxClients
MPM 16 ServerLimit 25 ( Threa
MaxClients 16 ServerLi
```



## MAXMEMFREE

```
[:] free()  
[:] MaxMemFree KBytes  
[:] MaxMemFree 0  
[:]  
[:] MPM  
[:] beos, leader, mpm_netware, prefork, threadpool,  
worker, mpm_winnt
```

MaxMemFree

free()



## MaxRequestsPerChild

```
MaxRequestsPerChild
MaxRequestsPerChild number
MaxRequestsPerChild 10000
MPM
leader, mpm_netware, mpm_winnt, mpmt_os2,
perchild, prefork, threadpool, worker
```

### MaxRequestsPerChild

MaxRequestsPerChild 0

```
mpm_netware mpm_winnt 0
```

### MaxRequestsPerChild:

- ()
- 

```
KeepAlive
```





- MinSpareThreads
- StartServers



MinSpareThreads

```

:
: MinSpareThreads number
:
:
: MPM
: beos, leader, mpm_netware, mpmt_os2, perchild,
threadpool, worker

```

MPM

perchild MinSpareThreads 5  
NumServers 10 MinSpareThreads 5 50

worker, leader, threadpool MinSpareThrea

mpm\_netware MinSpareThreads 10 MPM

beos mpmt\_os2 mpm\_netware beos  
MinSpareThreads 1 mpmt\_os2 5

- MaxSpareThreads
- StartServers



## HTTPD

```
: ID
: PidFile filename
: PidFile logs/httpd.pid
:
: MPM
: beos, leader, mpm_winnt, mpmt_os2, perchild,
prefork, threadpool, worker
```

PidFile ID

```
PidFile /var/run/apache.pid
```

ErrorLog TransferLog  
PidFile ID

PidFile

```
Apache 2 apachectl ()
```



```

:| TCP receive buffer size
:| ReceiveBufferSize bytes
:| ReceiveBufferSize 0
:|
:| MPM
:| beos, leader, mpm_netware, mpm_winnt, mpmt_os2,
:| perchild, prefork, threadpool, worker

```

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



```
ScoreBoardFile
ScoreBoardFile file-path
ScoreBoardFile logs/apache_status
MPM
beos, leader, mpm_winnt, perchild, prefork,
threadpool, worker
```

Apache

Apache

```
ScoreBoardFile /var/run/apache_status
```

ScoreBoardFile

RAM

- [Apache](#)



```

:~ TCP
:~ SendBufferSize bytes
:~ SendBufferSize 0
:~
:~ MPM
:~ beos, leader, mpm_netware, mpm_winnt, mpmt_os2,
perchild, prefork, threadpool, worker

```

TCP

00S



ServerLimit

```
:  
: ServerLimit number  
:  
:  
: MPM  
: leader, perchild, prefork, threadpool, worker
```

prefork MPM Apache !  
) worker MPM Thre  
MaxClients

ServerLimit

Apache

prefork MPM MaxClients 256 ()  
MaxClients

worker, leader, threadpool MPM MaxClients  
ThreadsPerChild 16 ()  
ThreadsPerChild

perchild MPM NumServers 8 ()

```
ServerLimit 20000
```

- [Apache](#)



## StartServers

```
StartServers number
MPM
leader, mpmt_os2, prefork, threadpool, worker
```

### StartServers

```
MPM leader, threadpool, worker
3 prefork 5 mpmt_os2 2
```



## StartThreads

```
:  
: StartThreads number  
:  
:  
: MPM  
: beos, mpm_netware, perchild
```

perchild                    StartThreads 5

mpm\_netware                StartThreads 50

beos    StartThreads 10



THREAD LIMIT

```

:
: ThreadLimit number
:
:
: MPM
: leader, mpm_winnt, perchild, threadpool, worker
: Apache 2.0.41 mpm_winnt

```

Apache

ThreadsPerChild

ThreadLimit Thr

Apac

ThreadsPerChild

ThreadsPerChild

ThreadLimit mpm\_winnt 1920 64

```

ThreadLimit 20000 (mpm_winnt ThreadLimit 1500

```



## ThreadsPerChild

```
⋮  
⋮ ThreadsPerChild number  
⋮  
⋮  
⋮ MPM  
⋮ leader, mpm_winnt, threadpool, worker
```

MPM

mpm\_winnt ThreadsPerChild 64 25



```
#: ID
#: User unix-userid
#: User #-1
#:
#: MPM
#: leader, perchild, prefork, threadpool, worker
#: Apache 2.0
```

User ID  
root *Unix-userid*

#

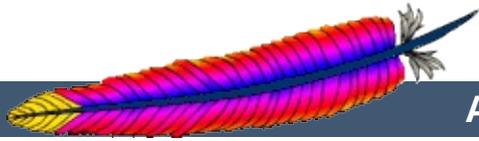
```
User ( Group)
```

perchild MPM ID  
<VirtualHost>

: <VirtualHost>

```
User beos mpmt_os2 MPM
```

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 beos

<b>Description:</b>	This Multi-Processing Module is optimized for BeOS.
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_beos_module
<b>Source File:</b>	beos.c

### Summary

This Multi-Processing Module (MPM) is the default for BeOS. It uses a single control process which creates threads to handle requests.

### See also

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



<b>Description:</b>	Limit on the number of requests that an individual thread will handle during its life
<b>Syntax:</b>	MaxRequestsPerThread <i>number</i>
<b>Default:</b>	MaxRequestsPerThread 0
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	beos

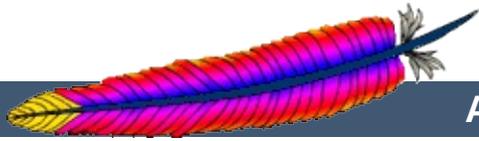
The `MaxRequestsPerThread` directive sets the limit on the number of requests that an individual server thread will handle. After `MaxRequestsPerThread` requests, the thread will die. If `MaxRequestsPerThread` is 0, then the thread will never expire.

Setting `MaxRequestsPerThread` to a non-zero limit has two beneficial effects:

- it limits the amount of memory that a thread can consume by (accidental) memory leakage;
- by giving threads a finite lifetime, it helps reduce the number of threads when the server load reduces.

**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 thread.



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



## Apache HTTP Server Version 2.0

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

# Apache MPM leader

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

## Summary

### Warning

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

This is an experimental variant of the standard [worker](#) MPM. It uses a Leader/Followers design pattern to coordinate work among threads. For more info, see <http://deuce.doc.wustl.edu/doc/pspdfs/lf.pdf>.

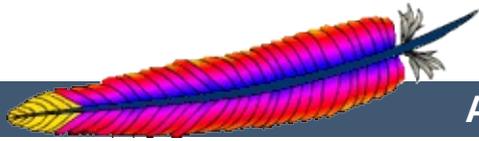
To use the [leader](#) MPM, add `--with-mpm=leader` to the [configure](#) script's arguments when building the [httpd](#).

This MPM depends on APR's atomic compare-and-swap operations for thread synchronization. If you are compiling for an x86 target and you don't need to support 386s, or you are compiling for a SPARC and you don't need to run on pre-UltraSPARC chips, add `--enable-nonportable-atomics=yes` to the [configure](#) script's arguments. This will cause APR to implement atomic operations using efficient opcodes not available in older CPUs.

---

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 netware

<b>Description:</b>	Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_netware_module
<b>Source File:</b>	mpm_netware.c

### Summary

This Multi-Processing Module (MPM) implements an exclusively threaded web server that has been optimized for Novell NetWare.

The main thread is responsible for launching child worker threads which listen for connections and serve them when they arrive. Apache always tries to maintain several *spare* or idle worker threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new child threads to be spawned before their requests can be served.

The [StartThreads](#), [MinSpareThreads](#), [MaxSpareThreads](#), and [MaxThreads](#) regulate how the main thread creates worker threads to serve requests. In general, Apache is very self-regulating, so most sites do not need to adjust these directives from their default values. Sites with limited memory may need to decrease [MaxThreads](#) to keep the server from thrashing (spawning and terminating idle threads). More information about tuning process creation is provided in the [performance hints](#) documentation.

[MaxRequestsPerChild](#) controls how frequently the server recycles processes by killing old ones and launching new ones. On the NetWare OS it is highly recommended that this directive remain set to 0. This allows worker threads to continue servicing requests

indefinitely.

## See also

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



## MaxThreads Directive

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

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

```
MaxThreads 512
```



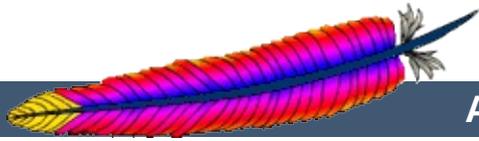
<b>Description:</b>	Determine the stack size for each thread
<b>Syntax:</b>	ThreadStackSize <i>number</i>
<b>Default:</b>	ThreadStackSize 65536
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	mpm_netware

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

<b>Description:</b>	Hybrid multi-process, multi-threaded MPM for OS/2
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_mpmt_os2_module
<b>Source File:</b>	mpmt_os2.c

## Summary

The Server consists of a main, parent process and a small, static number of child processes.

The parent process's job is to manage the child processes. This involves spawning children as required to ensure there are always [StartServers](#) processes accepting connections.

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

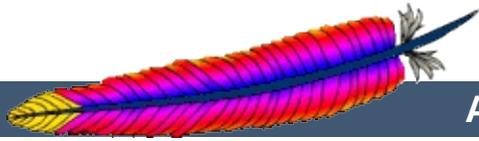
## See also

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

---

Copyright 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

<b>Description:</b>	Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_perchild_module
<b>Source File:</b>	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>
```



<b>Description:</b>	Tie a virtual host to a user and group ID
<b>Syntax:</b>	AssignUserID <i>user-id group-id</i>
<b>Context:</b>	virtual host
<b>Status:</b>	MPM
<b>Module:</b>	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](#).



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

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



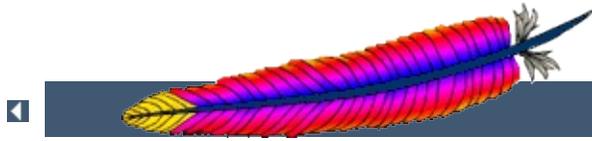
<b>Description:</b>	Total number of children alive at the same time
<b>Syntax:</b>	NumServers <i>number</i>
<b>Default:</b>	NumServers 2
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	perchild

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](#)



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache MPM prefork

This translation may be out of date. Check the English version for recent changes.

```
┆ fork
┆ MPM
┆ mpm_prefork_module
┆ prefork.c
```

(MPM)  
MPM

Unix Apache 1.3

MPM MPM

[Apache](#)



---

listen

StartSe

MaxClients Apache

)

Unix 80  
Apache

root

MaxRequestsPerChild



## maxspareservers

```
MaxSpareServers number
MaxSpareServers 10
MPM
prefork
```

MaxSpareServers  
kill

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

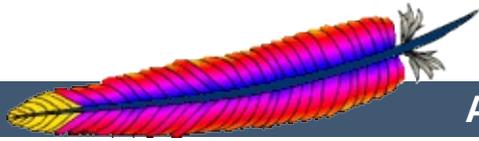


```
MinSpareServers number
MinSpareServers 5
MPM
prefork
```

### MaxSpareServers

1 1

- [MaxSpareServers](#)
- [StartServers](#)



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



## Apache HTTP Server Version 2.0

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

# Apache MPM threadpool

<b>Description:</b>	Yet another experimental variant of the standard <a href="#">worker</a> MPM
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_threadpool_module
<b>Source File:</b>	threadpool.c

## Summary

### Warning

This MPM is a developer playground and highly experimental, so it may or may not work as expected.

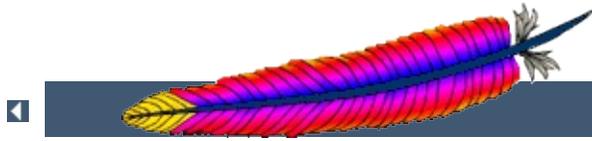
This is an experimental variant of the standard [worker](#) MPM. Rather than queuing connections like the [worker](#) MPM, the [threadpool](#) MPM queues idle worker threads and hands each accepted connection to the next available worker.

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](#)



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

## Apache MPM winnt

This translation may be out of date. Check the English version for recent changes.

- [: Windows NT](#)
- [: MPM](#)
- [: mpm\\_winnt\\_module](#)
- [: mpm\\_winnt.c](#)

(MPM)

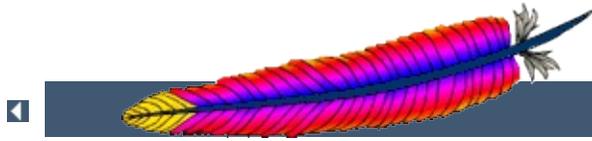
Windows NT



- [: accept\(\) AcceptEx](#)
- [: Win32DisableAcceptEx](#)
- [: AcceptEx\(\) AcceptEx\(\)](#)
- [:](#)
- [: MPM](#)
- [: mpm\\_winnt](#)
- [: 2.0.49](#)

AcceptEx() Microsoft WinSock v2 API BSD [a](#)  
 API Windows VPN  
 AcceptEx()

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache MPM worker

This translation may be out of date. Check the English version for recent changes.

```
┆  
┆ MPM  
┆ mpm_worker_module  
┆ worker.c
```

(MPM)

MPM  
ThreadsPerChild

ThreadsPerChild  
MaxClients

Apache



() [ThreadsPerC](#)

Apache

[MinSpareThreads](#) [MaxSpareThreads](#) fork

[ThreadsPerChild](#)

[ThreadsPerChild](#) [ThreadLim](#)

[ThreadsPerChild](#)

- [MaxRequestsPerChild](#) 0
- [MaxSpareThreads](#) [MaxClients](#)

[worker](#) MPM

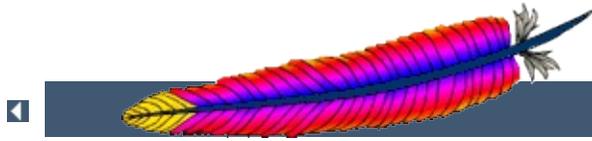
```
ServerLimit 16
StartServers 2
MaxClients 150
MinSpareThreads 25
MaxSpareThreads 75
ThreadsPerChild 25
```

Unix 80

root

Apache

[MaxRequestsPerChild](#)



| [FAQ](#) |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_access

This translation may be out of date. Check the English version for recent changes.

- [IP](#)
- [Base](#)
- [access\\_module](#)
- [mod\\_access.c](#)
- [2.1](#)

[mod\\_access](#)

[.htaccess](#) IP

[Order](#)

[Allow](#)

[Deny](#)

(GET, PUT, POST)

[Satisfy](#)

[Require](#)



## ALLOW

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

### Allow

from Allow from :

()

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

Apache [HostnameLookups](#) IP  
IP DNS

IP

```
:  
: Allow from 10.1.2.3
```

IP

IP

```
:
```

```
Allow from 10.1
```

IP

/

```
:
```

```
Allow from 10.1.0.0/255.255.0.0
```

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

**/nnn CIDR**

```
:
```

```
Allow from 10.1.0.0/16
```

nnn 1

:

**IPv6 IPv6 :**

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

**Allow**

*variable*

mod\_se

) **Referer HTTP**

```
:
```

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

user-agent

KnockKnock/2.0



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

IP



```

: Allow Deny
: Order ordering
: Order Deny,Allow
: , .htaccess
: Limit
: Base
: mod_access

```

Order Allow Deny

**Deny, Allow**

Deny Allow

**Allow, Deny**

Allow Deny

**Mutual-failure**

Allow Deny

Allow Deny

apache.org

```

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

```

foo.apache.org

apache.org

```

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

```

**Order** Deny, Allow

apache.org

Deny from foo.apache.org

apache.org

*allow*

Order

Allow

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

*deny*

*/www*

Order

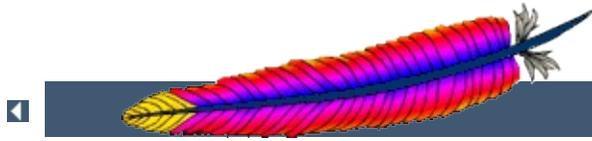
Directory

*.htaccess*

Allow

Den

[Directory,Location, Files](#)



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_actions

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

Action

CGI

mod\_cgi

CGI

Apache



```
CGI
Action action-type cgi-script
, , , .htaccess
FileInfo
Base
mod_actions
```

*action-type* *cgi-script*  
AddHandler CGI URL-path  
MIME URL

C

```
# Requests for files of a particular type:
Action image/gif /cgi-bin/images.cgi

# Files of a particular file extension
AddHandler my-file-type .xyz
Action my-file-type /cgi-bin/program.cgi
```

MIME image/gif

2 .xyz

- AddHandler



## Script

```
Script CGI
Script method cgi-script
Base
mod_actions
```

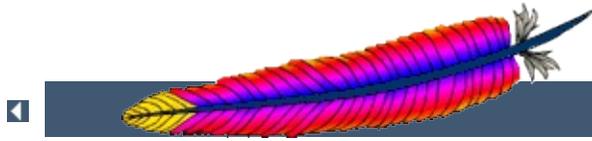
*method* *cgi-script*  
**AddHandler** CGI URL-path  
PATH\_INFO PATH\_TRANSLATED

Script PUT Scri

Script CGI

```
# For <ISINDEX>-style searching
Script GET /cgi-bin/search

# A CGI PUT handler
Script PUT /~bob/put.cgi
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_alias

This translation may be out of date. Check the English version for recent changes.

- [:](#)
- [Base](#)
- [alias\\_module](#)
- [mod\\_alias.c](#)

[URL](#)

[ScriptAlias](#) CGI

[Redirect](#)

[URL](#)

[mod\\_alias](#) [URL](#)

[mod\\_rewrite](#)

[URL](#)



```
Alias Redirect      (      <VirtualHost>
) Alias Redirect
```

```
Alias Redirect      Redirect  RedirectMatch
Alias Alias Redirect
```

:

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

```
/foo Alias      /foo/bar Alias
```



```
: URL
: Alias URL-path file-path|directory-path
: ,
: Base
: mod_alias
```

Alias

DocumentRoot

*directory-filename*

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

http://myserver/image/foo.gif

/ftp/pub/image

```
url-path / /
/usr/local/apache/icons/
```

/icons

<Directory>

( <Location>

Alias DocumentRoot

```
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

UR

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



## Redirect

```
: URL
: Redirect [status] URL-path URL
: , , , .htaccess
: FileInfo
: Base
: mod_alias
```

Redirect URL URL  
*URL* (%) *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)  
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 [status] regex URL  
: , , , .htaccess  
: FileInfo  
: Base  
: mod_alias
```

### Redirect

JPEG :

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```



## Redirect Options

```
: URL
: RedirectPermanent URL-path URL
: , , .htaccess
: FileInfo
: Base
: mod_alias
```

Redirect

( 301)



## RedirectTemp

```
: URL
: RedirectTemp URL-path URL
: , , , .htaccess
: FileInfo
: Base
: mod_alias
```

Redirect

( 302)



## ScriptAlias

```
: URL CGI
: ScriptAlias URL-path file-path|directory-path
: ,
: Base
: mod_alias
```

**ScriptAlias** *URL-path* mod\_cgi cgi-script  
URL (%) *URL-path*

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

http://myserver/cgi-bin/foo

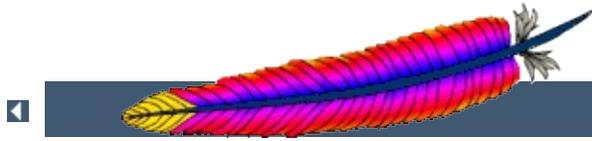


```
: URL CGI
: ScriptAliasMatch regex file-path|directory-
  path
: ,
: Base
: mod_alias
```

### ScriptAlias

bin :

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_asis

This translation may be out of date. Check the English version for recent changes.

- [: HTTP](#)
- [: Base](#)
- [: asis\\_module](#)
- [: mod\\_asis.c](#)

send-as-is HTTP

Cgi nph

mime

httpd/send-as-is

[mod\\_headers](#)

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

[Apache](#)



## send-as-is

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

.asis Apache

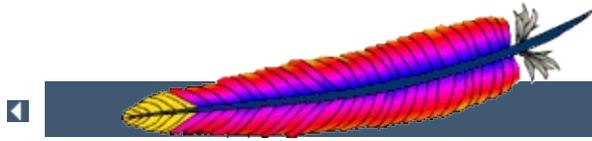
HTTP

*as is ()*

```
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:
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_auth

This translation may be out of date. Check the English version for recent changes.

- [:](#)
- [:](#) Base
- [:](#) auth\_module
- [:](#) mod\_auth.c
- [:](#) 2.1

HTTP

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

[Require](#)  
[Satisfy](#)  
[AuthName](#)  
[AuthType](#)



```

:
: AuthAuthoritative On|Off
: AuthAuthoritative On
: , .htaccess
: AuthConfig
: Base
: mod_auth

```

AuthAuthoritative Off ID  
(Configuration modules.c)  
"Authentication Required"

ID

mod\_auth\_dbm, mod\_auth\_sql, mod\_auth\_anon

() AuthUserFile

ID "Authenticator  
NCSA

```

        .htaccess
AuthUserFile AuthGroupFile
AuthUserFile AuthGroupFile

```



## AuthGroupFile

```
:  
: AuthGroupFile file-path  
: , .htaccess  
: AuthConfig  
: Base  
: mod_auth
```

### AuthGroupFile

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

### AuthDBMGroupF:

```
AuthGroupFile
```



## AuthUserFile

```
AuthUserFile file-path  
, .htaccess  
AuthConfig  
Base  
mod_auth
```

### AuthUserFile

ID username Filename :  
src/support ht

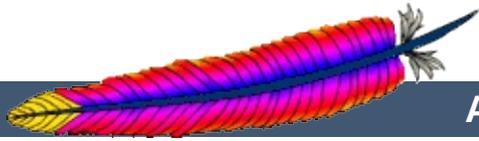
```
htpasswd -c Filename username
```

```
Filename username2 :
```

```
htpasswd Filename username2
```

(:)

### AuthUserFile



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



## Apache HTTP Server Version 2.0

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

## Apache Module mod\_auth\_anon

<b>Description:</b>	Allows "anonymous" user access to authenticated areas
<b>Status:</b>	Extension
<b>Module Identifier:</b>	auth_anon_module
<b>Source File:</b>	mod_auth_anon.c
<b>Compatibility:</b>	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>
```



<b>Description:</b>	Specifies userIDs that are allowed access without password verification
<b>Syntax:</b>	Anonymous <i>user</i> [ <i>user</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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

<b>Description:</b>	Configures if authorization will fall-through to other methods
<b>Syntax:</b>	Anonymous_Authoritative On Off
<b>Default:</b>	Anonymous_Authoritative Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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

<b>Description:</b>	Sets whether the password entered will be logged in the error log
<b>Syntax:</b>	Anonymous_LogEmail On Off
<b>Default:</b>	Anonymous_LogEmail On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

When set On, the default, the 'password' entered (which hopefully contains a sensible email address) is logged in the error log.



<b>Description:</b>	Specifies whether blank passwords are allowed
<b>Syntax:</b>	Anonymous_MustGiveEmail On Off
<b>Default:</b>	Anonymous_MustGiveEmail On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

Specifies whether the user must specify an email address as the password. This prohibits blank passwords.



<b>Description:</b>	Sets whether the userID field may be empty
<b>Syntax:</b>	Anonymous_NoUserID On Off
<b>Default:</b>	Anonymous_NoUserID Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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.



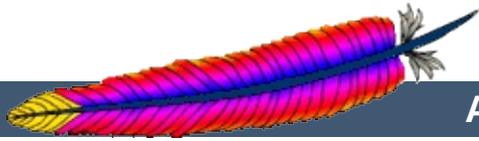
<b>Description:</b>	Sets whether to check the password field for a correctly formatted email address
<b>Syntax:</b>	Anonymous_VerifyEmail On Off
<b>Default:</b>	Anonymous_VerifyEmail Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

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

<b>Description:</b>	Provides for user authentication using DBM files
<b>Status:</b>	Extension
<b>Module Identifier:</b>	auth_dbm_module
<b>Source File:</b>	mod_auth_dbm.c
<b>Compatibility:</b>	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](#)



<b>Description:</b>	Sets whether authentication and authorization will be passed on to lower level modules
<b>Syntax:</b>	AuthDBMAuthoritative On Off
<b>Default:</b>	AuthDBMAuthoritative On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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.



<b>Description:</b>	Sets the name of the database file containing the list of user groups for authentication
<b>Syntax:</b>	<code>AuthDBMGroupFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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](http://www.telescope.org) uses for its combined password and group database.



## AuthDBMType Directive

<b>Description:</b>	Sets the type of database file that is used to store passwords
<b>Syntax:</b>	AuthDBMType default   SDBM   GDBM   NDBM   DB
<b>Default:</b>	AuthDBMType default
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_dbm
<b>Compatibility:</b>	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.



<b>Description:</b>	Sets the name of a database file containing the list of users and passwords for authentication
<b>Syntax:</b>	<code>AuthDBMUserFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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.

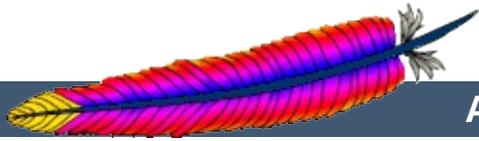
A perl script called `dbmmanage` is included with Apache. This

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](#)



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



## Apache HTTP Server Version 2.0

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

# Apache Module mod\_auth\_digest

<b>Description:</b>	User authentication using MD5 Digest Authentication.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	auth_digest_module
<b>Source File:</b>	mod_auth_digest.c

## Summary

This module implements HTTP Digest Authentication. However, it has not been extensively tested and is therefore marked experimental.

## See also

[AuthName](#)  
[AuthType](#)  
[Require](#)  
[Satisfy](#)



## Using Digest Authentication

Using MD5 Digest authentication is very simple. Simply set up authentication normally, using `AuthType Digest` and [AuthDigestFile](#) instead of the normal `AuthType Basic` and [AuthUserFile](#); also, replace any [AuthGroupFile](#) with [AuthDigestGroupFile](#). Then add a [AuthDigestDomain](#) directive containing at least the root URI(s) for this protection space.

Appropriate user (text) files can be created using the [htdigest](#) tool.

### Example:

```
<Location /private/>
  AuthType Digest
  AuthName "private area"
  AuthDigestDomain /private/ http://mirror.my.dom/private2/
  AuthDigestFile /web/auth/.digest_pw
  Require valid-user
</Location>
```

### Note

Digest authentication provides a more secure password system than Basic authentication, but only works with supporting browsers. As of November 2002, the major browsers that support digest authentication are [Opera](#), [MS Internet Explorer](#) (fails when used with a query string - see "[Working with MS Internet Explorer](#)" below for a workaround), [Amaya](#), [Mozilla](#) and [Netscape](#) since version 7. Since digest authentication is not as widely implemented as basic authentication, you should use it only in controlled environments.



The Digest authentication implementation in previous Internet Explorer for Windows versions (5 and 6) had issues, namely that GET requests with a query string were not RFC compliant. There are a few ways to work around this issue.

The first way is to use POST requests instead of GET requests to pass data to your program. This method is the simplest approach if your application can work with this limitation.

Since version 2.0.51 Apache also provides a workaround in the `AuthDigestEnableQueryStringHack` environment variable. If `AuthDigestEnableQueryStringHack` is set for the request, Apache will take steps to work around the MSIE bug and remove the query string from the digest comparison. Using this method would look similar to the following.

### Using Digest Authentication with MSIE:

```
BrowserMatch "MSIE" AuthDigestEnableQueryStringHack=On
```

This workaround is not necessary for MSIE 7, though enabling it does not cause any compatibility issues or significant overhead.

See the [BrowserMatch](#) directive for more details on conditionally setting environment variables



## AuthDigestAlgorithm Directive

<b>Description:</b>	Selects the algorithm used to calculate the challenge and response hashes in digest authentication
<b>Syntax:</b>	<code>AuthDigestAlgorithm MD5 MD5-sess</code>
<b>Default:</b>	<code>AuthDigestAlgorithm MD5</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestAlgorithm` directive selects the algorithm used to calculate the challenge and response hashes.

MD5-sess is not correctly implemented yet.



<b>Description:</b>	URIs that are in the same protection space for digest authentication
<b>Syntax:</b>	AuthDigestDomain <i>URI</i> [ <i>URI</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestDomain` directive allows you to specify one or more URIs which are in the same protection space (*i.e.* use the same realm and username/password info). The specified URIs are prefixes, *i.e.* the client will assume that all URIs "below" these are also protected by the same username/password. The URIs may be either absolute URIs (*i.e.* including a scheme, host, port, etc) or relative URIs.

This directive *should* always be specified and contain at least the (set of) root URI(s) for this space. Omitting to do so will cause the client to send the Authorization header for *every request* sent to this server. Apart from increasing the size of the request, it may also have a detrimental effect on performance if `AuthDigestNcCheck` is on.

The URIs specified can also point to different servers, in which case clients (which understand this) will then share username/password info across multiple servers without prompting the user each time.



<b>Description:</b>	Location of the text file containing the list of users and encoded passwords for digest authentication
<b>Syntax:</b>	<code>AuthDigestFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestFile` directive sets the name of a textual file containing the list of users and encoded passwords for digest authentication. *File-path* is the absolute path to the user file.

The digest file uses a special format. Files in this format can be created using the `htdigest` utility found in the `support/` subdirectory of the Apache distribution.



## AuthDigestGroupFile Directive

<b>Description:</b>	Name of the text file containing the list of groups for digest authentication
<b>Syntax:</b>	<code>AuthDigestGroupFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestGroupFile` directive sets the name of a textual file containing the list of groups and their members (user names). *File-path* is the absolute path to the group file.

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.

**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`.



## AuthDigestNcCheck Directive

**Description:** Enables or disables checking of the nonce-count sent by the server

**Syntax:** AuthDigestNcCheck On|Off

**Default:** AuthDigestNcCheck Off

**Context:** server config

**Status:** Experimental

**Module:** mod\_auth\_digest

Not implemented yet.



<b>Description:</b>	Determines how the nonce is generated
<b>Syntax:</b>	AuthDigestNonceFormat <i>format</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

Not implemented yet.



<b>Description:</b>	How long the server nonce is valid
<b>Syntax:</b>	<code>AuthDigestNonceLifetime seconds</code>
<b>Default:</b>	<code>AuthDigestNonceLifetime 300</code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestNonceLifetime` directive controls how long the server nonce is valid. When the client contacts the server using an expired nonce the server will send back a 401 with `stale=true`. If `seconds` is greater than 0 then it specifies the amount of time for which the nonce is valid; this should probably never be set to less than 10 seconds. If `seconds` is less than 0 then the nonce never expires.



<b>Description:</b>	Determines the quality-of-protection to use in digest authentication
<b>Syntax:</b>	AuthDigestQop none auth auth-int [auth auth-int]
<b>Default:</b>	AuthDigestQop auth
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestQop` directive determines the *quality-of-protection* to use. `auth` will only do authentication (username/password); `auth-int` is authentication plus integrity checking (an MD5 hash of the entity is also computed and checked); `none` will cause the module to use the old RFC-2069 digest algorithm (which does not include integrity checking). Both `auth` and `auth-int` may be specified, in which the case the browser will choose which of these to use. `none` should only be used if the browser for some reason does not like the challenge it receives otherwise.

`auth-int` is not implemented yet.



**Description:** The amount of shared memory to allocate for keeping track of clients

**Syntax:** AuthDigestShmemSize *size*

**Default:** AuthDigestShmemSize 1000

**Context:** server config

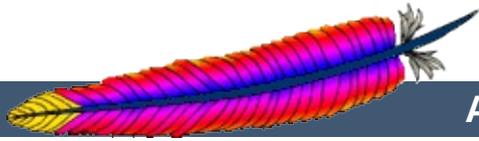
**Status:** Experimental

**Module:** mod\_auth\_digest

The `AuthDigestShmemSize` directive defines the amount of shared memory, that will be allocated at the server startup for keeping track of clients. Note that the shared memory segment cannot be set less than the space that is necessary for tracking at least *one* client. This value is dependant on your system. If you want to find out the exact value, you may simply set `AuthDigestShmemSize` to the value of 0 and read the error message after trying to start the server.

The *size* is normally expressed in Bytes, but you may let the number follow a K or an M to express your value as KBytes or MBytes. For example, the following directives are all equivalent:

```
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`

<b>Description:</b>	Allows an LDAP directory to be used to store the database for HTTP Basic authentication.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>auth_ldap_module</code>
<b>Source File:</b>	<code>mod_auth_ldap.c</code>
<b>Compatibility:</b>	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

<a href="#">AuthLDAPURL</a>	Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to
-----------------------------	---

	use.
<a href="#">AuthLDAPBindDN</a>	An optional DN to bind with during the search phase.
<a href="#">AuthLDAPBindPassword</a>	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:

<a href="#">AuthLDAPURL</a>	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.



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



<b>Description:</b>	Optional DN to use in binding to the LDAP server
<b>Syntax:</b>	AuthLDAPBindDN <i>distinguished-name</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Password used in conjunction with the bind DN
<b>Syntax:</b>	AuthLDAPBindPassword <i>password</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Language to charset conversion configuration file
<b>Syntax:</b>	<code>AuthLDAPCharsetConfig</code> <i>file-path</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<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.



<b>Description:</b>	Use the LDAP server to compare the DN's
<b>Syntax:</b>	AuthLDAPCompareDNOnServer on off
<b>Default:</b>	AuthLDAPCompareDNOnServer on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	When will the module de-reference aliases
<b>Syntax:</b>	AuthLDAPDereferenceAliases never   searching   finding   always
<b>Default:</b>	AuthLDAPDereferenceAliases Always
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

This directive specifies when `mod_auth_ldap` will de-reference aliases during LDAP operations. The default is `always`.



<b>Description:</b>	Turn on or off LDAP authentication
<b>Syntax:</b>	AuthLDAPEnabled on off
<b>Default:</b>	AuthLDAPEnabled on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Allow LDAP authentication to work with MS FrontPage
<b>Syntax:</b>	AuthLDAPFrontPageHack on off
<b>Default:</b>	AuthLDAPFrontPageHack off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

See the section on [using Microsoft FrontPage](#) with [mod\\_auth\\_ldap](#).



## AuthLDAPGroupAttribute Directive

<b>Description:</b>	LDAP attributes used to check for group membership
<b>Syntax:</b>	AuthLDAPGroupAttribute <i>attribute</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Use the DN of the client username when checking for group membership
<b>Syntax:</b>	AuthLDAPGroupAttributeIsDN on off
<b>Default:</b>	AuthLDAPGroupAttributeIsDN on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Use the DN of the client username to set the REMOTE_USER environment variable
<b>Syntax:</b>	AuthLDAPRemoteUserIsDN on off
<b>Default:</b>	AuthLDAPRemoteUserIsDN off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	URL specifying the LDAP search parameters
<b>Syntax:</b>	AuthLDAPUrl <i>url</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	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

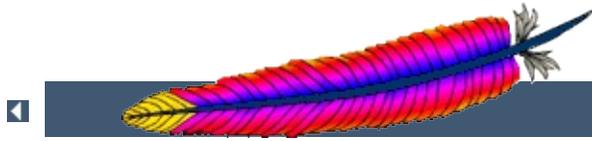
username of Babs Jenson, the resulting search filter will be (& (posixid=\*)(cn=Babs Jenson)).

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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_autoindex

This translation may be out of date. Check the English version for recent changes.

```
⌵ Unix  ls      Win32
  dir
⌵ Base
⌵ autoindex_module
⌵ mod_autoindex.c
```

:

- index.html [DirectoryIndex](#)
- [AddIconByType](#)

()

Options +Indexes

[Options](#)

[FancyIndexing](#)

[IndexOptions](#)

[IndexOptions](#)

SuppressColumnSorting

"Size"

-



## Apache 2.0.23

- C=N
- C=M
- C=S
- C=D
  
- O=A
- O=D
  
- F=0 (FancyIndex )
- F=1 FancyIndex
- F=2 HTML FancyIndex
- V=0
- V=1
  
- P=*pattern* *pattern*

"P ( P)" [IndexIgnore](#)  
[mod\\_autoindex](#) ()

HEADER.html

```
<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>
```



```
:\  
:\ AddAlt string file [file] ...  
:\ , , , .htaccess  
:\ Indexes  
:\ Base  
:\ mod_autoindex
```

AddAlt FancyIndexing  
(" ')

```
AddAlt "PDF file" *.pdf  
AddAlt Compressed *.gz *.zip *.Z
```



- [: MIME](#)
- [: AddAltByEncoding \*string MIME-encoding \[MIME-encoding\] ...\*](#)
- [: , , , .htaccess](#)
- [: Indexes](#)
- [: Base](#)
- [: mod\\_autoindex](#)

[AddAltByEncoding](#)      [FancyIndexing](#)  
*encoding*                      x-compress              *string ( " ' )*

```
AddAltByEncoding gzip x-gzip
```



## AddAltByType

```
: MIME
: AddAltByType string MIME-type [MIME-type] ...
: , , , .htaccess
: Indexes
: Base
: mod_autoindex
```

```
AddAltByType FancyIndexing
text/html string ( " ')
```

```
AddAltByType 'plain text' text/plain
```



## AddDescription

```
:  
: AddDescription string file [file] ...  
: , , , .htaccess  
: Indexes  
: Base  
: mod_autoindex
```

### [FancyIndexing](#)

*file*

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

23 [IndexOptions Suppress](#)  
[IndexOptions SuppressSize](#) 7 [IndexOpt](#)  
[SuppressLastModified](#) 19

```
AddDescription HTML
```



## BLANKICON

```
:\n:\nAddIcon icon name [name] ...\n:\n, , , .htaccess\n:\nIndexes\n:\nBase\n:\nmod_autoindex
```

### FancyIndexing

(*alttext*, *url*)

*name*

*alttext*

*name*

^^DIRECTORY^^

^^BLANKICON^^ (

```
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm\nAddIcon /icons/dir.xbm ^^DIRECTORY^^\nAddIcon /icons/backup.xbm *~
```

AddIcon AddIconByType



## AddIconByEncoding

- [MIME](#)
- [AddIconByEncoding \*icon MIME-encoding \[MIME-encoding\] ...\*](#)
- [,,, .htaccess](#)
- [Indexes](#)
- [Base](#)
- [mod\\_autoindex](#)

### [FancyIndexing](#)

*icor.*

*(alttext, url)*

*alttext*

*MIME-encoding*

```
AddIconByEncoding /icons/compress.xbm x-compress
```



## AddIconByType

```
⋮ MIME
⋮ AddIconByType icon MIME-type [MIME-type] ...
⋮ , , , .htaccess
⋮ Indexes
⋮ Base
⋮ mod_autoindex
```

### FancyIndexing

*icon*

*(alttext, url)*

*alttext*

*MIME-type*

```
AddIconByType (IMG,/icons/image.xbm) image/*
```



## DefaultIcon

```
DefaultIcon url-path
DefaultIcon , , , .htaccess
DefaultIcon Indexes
DefaultIcon Base
DefaultIcon mod_autoindex
```

## FancyIndexing

```
DefaultIcon /icon/unknown.xbm
```



## HeaderName

```
HeaderName filename  
HeaderName , , , .htaccess  
HeaderName Indexes  
HeaderName Base  
HeaderName mod_autoindex
```

### HeaderName

```
HeaderName HEADER.html
```

```
HeaderName ReadmeName filename URI  
filename DocumentRoot
```

```
HeaderName /include/HEADER.html
```

```
filename " text/*" (text/html, text/  
CGI
```

```
AddType text/html .cgi
```

```
Options MultiViews file  
text/html options Includes Include  
(mod_include)
```

```
HeaderName HTML (<html>, <head>,  
IndexOptions +SuppressHTMLPreamble
```



## IndexIgnore

```
IndexIgnore file [file] ...  
,,, .htaccess  
Indexes  
Base  
mod_autoindex
```

### IndexIgnore

()

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



```

:
: IndexOptions [+|-]option [[+|-]option] ...
: , , .htaccess
: Indexes
: Base
: mod_autoindex

```

**IndexOptions** *option :*

**DescriptionWidth=[n | \*] (2.0.23 )**

DescriptionWidth  
 -DescriptionWidth () mod\_autoindex  
 DescriptionWidth=*n* *n*  
 DescriptionWidth=\*

**AddDescription**

**FancyIndexing**

**FoldersFirst (2.0.23 )**

Zed	Beta	Gamma
-----	------	-------

**HTMLTable ( Apache 2.0.23 )**

FancyIndexing

**IconsAreLinks**

FancyIndexing

**IconHeight[=*pixels*]**

IconWidth  
 Apache

**IconWidth[=*pixels*]**

IconHeight

**IgnoreCase**

gamma )

**IgnoreClient**

mod\_autoindex

SuppressColumnSorting )

**NameWidth=[n | \*]**

NameWidth

-NameWidth ()

mod\_autoindex

NameWidth=n

n

NameWidth=\*

**ScanHTMLTitles**

FancyIndexing

HTML

httpd title

CPU (

**SuppressColumnSorting**

Apache

FancyIndexing

2.0.23

IgnoreClient

**SuppressDescription**

FancyIndexing

AddDescription

De

**SuppressHTMLPreamble**

HeaderName

HTML

SuppressHTMLPreamble

**SuppressIcon (Apache 2.0.23 )**

FancyIndexing

SuppressIco

HTML 3.2

HTML 3.2

(FancyIndexing )

## SuppressLastModified

FancyIndexing

## SuppressRules (Apache 2.0.23 )

( hr )

SuppressIcon

SuppressRules

HTML 3.2

HTML 3.2

(FancyIndexing )

## SuppressSize

FancyIndexing

## TrackModified (Apache 2.0.23 )

HTTP

ETag

OS2 JFS Win32 NTFS

OS2 W

HEAD

## VersionSort (Apache 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 (Apache 2.0.49 )

XHTML [mod\\_autoindex](#) HTML 3.2 XHTML 1.0

## IndexOptions

## Apache 1.3.3

## IndexOptions

- **IndexOptions**

```
<Directory /foo>
  IndexOptions HTMLTable
  IndexOptions SuppressColumnSorting
</Directory>
```

```
IndexOptions HTMLTable SuppressColumnSorting
```

- ('+' '-' )

'+' '-'

**IndexOp**

```
IndexOptions +ScanHTMLTitles -IconsAreLinks FancyIndexing
IndexOptions +SuppressSize
```

IndexOptions FancyIndexing +SuppressSize  
FancyIndexing

**IndexOptions**



## INDEXORDERDEFAULT

```
:  
: IndexOrderDefault Ascending|Descending  
  Name|Date|Size|Description  
: IndexOrderDefault Ascending Name  
: , , , .htaccess  
: Indexes  
: Base  
: mod_autoindex
```

IndexOrderDefault

[FancyIndexing](#)

IndexOrderDefault

IndexOrderDefault

Name, Date, Size Description

[SuppressColumnSorting](#)



## HeaderName

```
ReadmeName  
ReadmeName filename  
, , , .htaccess  
Indexes  
Base  
mod_autoindex
```

ReadmeName

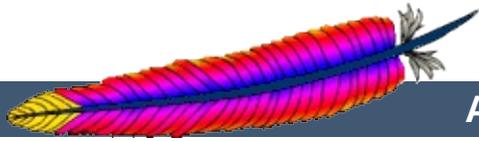
DocumentRoot

```
ReadmeName FOOTER.html
```

**2**

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

HeaderName



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



## Apache HTTP Server Version 2.0

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

# Apache Module `mod_cache`

<b>Description:</b>	Content cache keyed to URIs.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>cache_module</code>
<b>Source File:</b>	<code>mod_cache.c</code>

## Summary

This module is experimental. Documentation is still under development...

`mod_cache` implements an [RFC 2616](#) compliant HTTP content cache that can be used to cache either local or proxied content. `mod_cache` requires the services of one or more storage management modules. Two storage management modules are included in the base Apache distribution:

### `mod_disk_cache`

implements a disk based storage manager.

### `mod_mem_cache`

implements a memory based storage manager. `mod_mem_cache` can be configured to operate in two modes: caching open file descriptors or caching objects in heap storage. `mod_mem_cache` can be used to cache locally generated content or to cache backend server content for `mod_proxy` when configured using [ProxyPass](#) (aka *reverse proxy*)

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.



Related Modules	Related Directives
<a href="#">mod disk cache</a>	<a href="#">CacheRoot</a>
<a href="#">mod mem cache</a>	<a href="#">CacheSize</a>
	<a href="#">CacheGcInterval</a>
	<a href="#">CacheDirLevels</a>
	<a href="#">CacheDirLength</a>
	<a href="#">CacheExpiryCheck</a>
	<a href="#">CacheMinFileSize</a>
	<a href="#">CacheMaxFileSize</a>
	<a href="#">CacheTimeMargin</a>
	<a href="#">CacheGcDaily</a>
	<a href="#">CacheGcUnused</a>
	<a href="#">CacheGcClean</a>
	<a href="#">CacheGcMemUsage</a>
	<a href="#">MCacheSize</a>
	<a href="#">MCacheMaxObjectCount</a>
	<a href="#">MCacheMinObjectSize</a>
	<a href="#">MCacheMaxObjectSize</a>
	<a href="#">MCacheRemovalAlgorithm</a>
	<a href="#">MCacheMaxStreamingBuffer</a>



## Sample httpd.conf

```
#
# Sample Cache Configuration
#
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 Directive

<b>Description:</b>	The default duration to cache a document when no expiry date is specified.
<b>Syntax:</b>	CacheDefaultExpire <i>seconds</i>
<b>Default:</b>	CacheDefaultExpire 3600 (one hour)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

The `CacheDefaultExpire` directive specifies a default time, in seconds, to cache a document if neither an expiry date nor last-modified date are provided with the document. The value specified with the `CacheMaxExpire` directive does *not* override this setting.

```
CacheDefaultExpire 86400
```



## CACHE\_DISABLE\_DIRECTIVE

<b>Description:</b>	Disable caching of specified URLs
<b>Syntax:</b>	CacheDisable <i>url-string</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

The `CacheDisable` directive instructs `mod_cache` to *not* cache urls at or below *url-string*.

### Example

```
CacheDisable /local_files
```



<b>Description:</b>	Enable caching of specified URLs using a specified storage manager
<b>Syntax:</b>	<code>CacheEnable <i>cache_type url-string</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_cache</code>

The `CacheEnable` directive instructs `mod_cache` to cache urls at or below `url-string`. The cache storage manager is specified with the `cache_type` argument. `cache_type mem` instructs `mod_cache` to use the memory based storage manager implemented by `mod_mem_cache`. `cache_type disk` instructs `mod_cache` to use the disk based storage manager implemented by `mod_disk_cache`. `cache_type fd` instructs `mod_cache` to use the file descriptor cache implemented by `mod_mem_cache`.

In the event that the URL space overlaps between different `CacheEnable` directives (as in the example below), each possible storage manager will be run until the first one that actually processes the request. The order in which the storage managers are run is determined by the order of the `CacheEnable` directives in the configuration file.

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



**Description:** Percentage of document served, after which the server will complete caching the file even if the request is cancelled.

**Syntax:** `CacheForceCompletion Percentage`

**Default:** `CacheForceCompletion 60`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_cache`

Ordinarily, if a request is cancelled while the response is being cached and delivered to the client the processing of the response will stop and the cache entry will be removed. The `CacheForceCompletion` directive specifies a threshold beyond which the document will continue to be cached to completion, even if the request is cancelled.

The threshold is a percentage specified as a value between 1 and 100. A value of 0 specifies that the default be used. A value of 100 will only cache documents that are served in their entirety. A value between 60 and 90 is recommended.

```
CacheForceCompletion 80
```

**Note:**

This feature is currently *not* implemented.



<b>Description:</b>	Ignore the fact that the client requested the content not be cached.
<b>Syntax:</b>	CacheIgnoreCacheControl On Off
<b>Default:</b>	CacheIgnoreCacheControl Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

Ordinarily, documents with no-cache or no-store header values will not be stored in the cache. The `CacheIgnoreCacheControl` directive allows this behavior to be overridden.

`CacheIgnoreCacheControl On` tells the server to attempt to cache the document even if it contains no-cache or no-store header values. Documents requiring authorization will *never* be cached.

```
CacheIgnoreCacheControl On
```



## CacheIgnoreHeaders Directive

<b>Description:</b>	Do not store the given HTTP header(s) in the cache.
<b>Syntax:</b>	CacheIgnoreHeaders <i>header-string</i> [ <i>header-string</i> ] ...
<b>Default:</b>	CacheIgnoreHeaders None
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

According to RFC 2616, hop-by-hop HTTP headers are not stored in the cache. The following HTTP headers are hop-by-hop headers and thus do not get stored in the cache in *any* case regardless of the setting of **CacheIgnoreHeaders**:

- Connection
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade

**CacheIgnoreHeaders** specifies additional HTTP headers that should not to be stored in the cache. For example, it makes sense in some cases to prevent cookies from being stored in the cache.

**CacheIgnoreHeaders** takes a space separated list of HTTP headers that should not be stored in the cache. If only hop-by-hop headers not should be stored in the cache (the RFC 2616 compliant behaviour), **CacheIgnoreHeaders** can be set to None.

## Example 1

CacheIgnoreHeaders Set-Cookie

## Example 2

CacheIgnoreHeaders None

### Warning:

If headers like Expires which are needed for proper cache management are not stored due to a `CacheIgnoreHeaders` setting, the behaviour of `mod_cache` is undefined.



## CacheIgnoreNoLastMod Directive

<b>Description:</b>	Ignore the fact that a response has no Last Modified header.
<b>Syntax:</b>	CacheIgnoreNoLastMod On Off
<b>Default:</b>	CacheIgnoreNoLastMod Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

Ordinarily, documents without a last-modified date are not cached. Under some circumstances the last-modified date is removed (during `mod_include` processing for example) or not provided at all. The `CacheIgnoreNoLastMod` directive provides a way to specify that documents without last-modified dates should be considered for caching, even without a last-modified date. If neither a last-modified date nor an expiry date are provided with the document then the value specified by the `CacheDefaultExpire` directive will be used to generate an expiration date.

```
CacheIgnoreNoLastMod On
```



**Description:** The factor used to compute an expiry date based on the LastModified date.

**Syntax:** CacheLastModifiedFactor *float*

**Default:** CacheLastModifiedFactor 0.1

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_cache

In the event that a document does not provide an expiry date but does provide a last-modified date, an expiry date can be calculated based on the time since the document was last modified. The `CacheLastModifiedFactor` directive specifies a *factor* to be used in the generation of this expiry date according to the following formula:  $\text{expiry-period} = \text{time-since-last-modified-date} * \text{factor}$   $\text{expiry-date} = \text{current-date} + \text{expiry-period}$  For example, if the document was last modified 10 hours ago, and *factor* is 0.1 then the expiry-period will be set to  $10 * 0.1 = 1$  hour. If the current time was 3:00pm then the computed expiry-date would be 3:00pm + 1hour = 4:00pm. If the expiry-period would be longer than that set by `CacheMaxExpire`, then the latter takes precedence.

```
CacheLastModifiedFactor 0.5
```

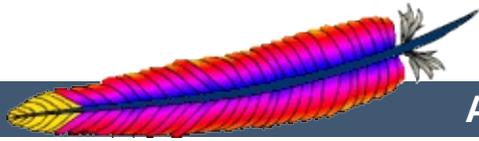


## CacheMaxExpire Directive

<b>Description:</b>	The maximum time in seconds to cache a document
<b>Syntax:</b>	CacheMaxExpire <i>seconds</i>
<b>Default:</b>	CacheMaxExpire 86400 (one day)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

The `CacheMaxExpire` directive specifies the maximum number of seconds for which cachable HTTP documents will be retained without checking the origin server. Thus, documents will be out of date at most this number of seconds. This maximum value is enforced even if an expiry date was supplied with the document.

```
CacheMaxExpire 604800
```



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



## Apache HTTP Server Version 2.0

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

# Apache Module mod\_cern\_meta

<b>Description:</b>	CERN httpd metafile semantics
<b>Status:</b>	Extension
<b>Module Identifier:</b>	cern_meta_module
<b>Source File:</b>	mod_cern_meta.c

## Summary

Emulate the CERN HTTPD Meta file semantics. Meta files are HTTP headers that can be output in addition to the normal range of headers for each file accessed. They appear rather like the Apache .asis files, and are able to provide a crude way of influencing the Expires: header, as well as providing other curiosities. There are many ways to manage meta information, this one was chosen because there is already a large number of CERN users who can exploit this module.

More information on the [CERN metafile semantics](#) is available.

## See also

[mod\\_headers](#)

[mod\\_asis](#)



<b>Description:</b>	Name of the directory to find CERN-style meta information files
<b>Syntax:</b>	MetaDir <i>directory</i>
<b>Default:</b>	MetaDir .web
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_cern_meta

Specifies the name of the directory in which Apache can find meta information files. The directory is usually a 'hidden' subdirectory of the directory that contains the file being accessed. Set to "." to look in the same directory as the file:

```
MetaDir .
```

Or, to set it to a subdirectory of the directory containing the files:

```
MetaDir .meta
```



<b>Description:</b>	Activates CERN meta-file processing
<b>Syntax:</b>	MetaFiles on off
<b>Default:</b>	MetaFiles off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_cern_meta

Turns on/off Meta file processing on a per-directory basis.



<b>Description:</b>	File name suffix for the file containing CERN-style meta information
<b>Syntax:</b>	MetaSuffix <i>suffix</i>
<b>Default:</b>	MetaSuffix .meta
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_cern_meta

Specifies the file name suffix for the file containing the meta information. For example, the default values for the two directives will cause a request to `DOCUMENT_ROOT/somedir/index.html` to look in `DOCUMENT_ROOT/somedir/.web/index.html.meta` and will use its contents to generate additional MIME header information.

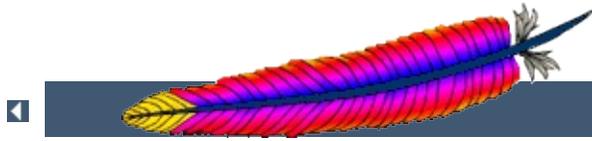
**Example:**

```
MetaSuffix .meta
```

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_cgi

- CGI
- Base
- cgi\_module
- mod\_cgi.c

Mime application/x-httpd-cgi                      cgi-script  
(Apache 1.1)      CGI

[ScriptAlias](#)

CGI

DOCUMENT\_ROOT

Apache CGI

[CGI](#)

Unix MPM

[mod\\_c](#)

[AcceptPathInfo](#)

[Options](#)

[ScriptAlias](#)

[AddHandler](#)

[CGI ID](#)

[CGI](#)



[CGI](#) CGI

### PATH\_INFO

[AcceptPathInfo](#) off

[mod\\_cgi](#)

(URI

/more

NOT FOUND

[AcceptPathInfo](#)

### REMOTE\_HOST

[HostnameLookups](#) on ( off )

DNS

### REMOTE\_IDENT

[IdentityCheck](#) on

ider

### REMOTE\_USER

CGI



CGI ()

**CGI**

CGI CGI CGI

```
%% [time] request-line
%% HTTP-status CGI-script-filename
```

CGI 2 :

```
%%error
error-message
```

)

```
%request
HTTP
() POST PUT
%response
CGI
%stdout
CGI
%stderr
CGI
```

( %stdout %stderr



```
ScriptLog CGI  
ScriptLog file-path  
,  
Base  
mod_cgi, mod_cgid
```

ScriptLog CGI  
ServerRoot

```
ScriptLog logs/cgi_log
```

User

CGI



## ScriptLogBuffer

```
: PUT POST
: ScriptLogBuffer bytes
: ScriptLogBuffer 1024
: ,
: Base
: mod_cgi, mod_cgid
```

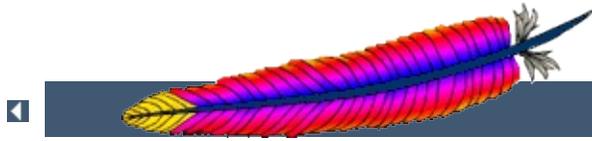
PUT POST



## ScriptLogLength

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

### ScriptLogLength CGI CGI



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >



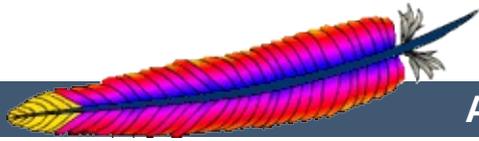
## ScriptSock

```
: CGI
: ScriptSock file-path
: ScriptSock logs/cgisock
: ,
: Base
: mod_cgid
```

CGI

Apache (root)

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



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



## Apache HTTP Server Version 2.0

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

## Apache Module `mod_charset_lite`

<b>Description:</b>	Specify character set translation or recoding
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>charset_lite_module</code>
<b>Source File:</b>	<code>mod_charset_lite.c</code>

### Summary

This is an **experimental** module and should be used with care. Experiment with your `mod_charset_lite` configuration to ensure that it performs the desired function.

`mod_charset_lite` allows the administrator to specify the source character set of objects as well as the character set they should be translated into before sending to the client. `mod_charset_lite` does not translate the data itself but instead tells Apache what translation to perform. `mod_charset_lite` is applicable to EBCDIC and ASCII host environments. In an EBCDIC environment, Apache normally translates text content from the code page of the Apache process locale to ISO-8859-1. `mod_charset_lite` can be used to specify that a different translation is to be performed. In an ASCII environment, Apache normally performs no translation, so `mod_charset_lite` is needed in order for any translation to take place.

This module provides a small subset of configuration mechanisms implemented by Russian Apache and its associated `mod_charset`.



## Invalid character set names

The character set name parameters of [CharsetSourceEnc](#) and [CharsetDefault](#) must be acceptable to the translation mechanism used by APR on the system where [mod\\_charset\\_lite](#) is deployed. These character set names are not standardized and are usually not the same as the corresponding values used in http headers. Currently, APR can only use `iconv(3)`, so you can easily test your character set names using the `iconv(1)` program, as follows:

```
iconv -f charsetsourceenc-value -t charsetdefault-value
```

## Mismatch between character set of content and translation rules

If the translation rules don't make sense for the content, translation can fail in various ways, including:

- The translation mechanism may return a bad return code, and the connection will be aborted.
- The translation mechanism may silently place special characters (e.g., question marks) in the output buffer when it cannot translate the input buffer.



<b>Description:</b>	Charset to translate into
<b>Syntax:</b>	CharsetDefault <i>charset</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Experimental
<b>Module:</b>	mod_charset_lite

The `CharsetDefault` directive specifies the charset that content in the associated container should be translated to.

The value of the *charset* argument must be accepted as a valid character set name by the character set support in APR. Generally, this means that it must be supported by iconv.

### Example

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



<b>Description:</b>	Configures charset translation behavior
<b>Syntax:</b>	CharsetOptions <i>option</i> [ <i>option</i> ] ...
<b>Default:</b>	CharsetOptions DebugLevel=0 NoImplicitAdd
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Experimental
<b>Module:</b>	mod_charset_lite

The `CharsetOptions` directive configures certain behaviors of `mod_charset_lite`. *Option* can be one of

### **DebugLevel=*n***

The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_charset_lite`. By default, no messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the `DBGLVL_` constants near the beginning of `mod_charset_lite.c`.

### **ImplicitAdd | NoImplicitAdd**

The `ImplicitAdd` keyword specifies that `mod_charset_lite` should implicitly insert its filter when the configuration specifies that the character set of content should be translated. If the filter chain is explicitly configured using the `AddOutputFilter` directive, `NoImplicitAdd` should be specified so that `mod_charset_lite` doesn't add its filter.



<b>Description:</b>	Source charset of files
<b>Syntax:</b>	CharsetSourceEnc <i>charset</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Experimental
<b>Module:</b>	mod_charset_lite

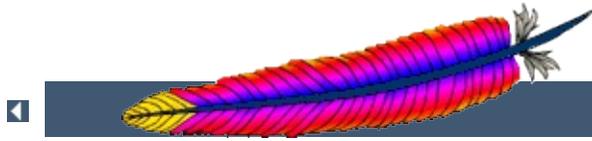
The **CharsetSourceEnc** directive specifies the source charset of files in the associated container.

The value of the *charset* argument must be accepted as a valid character set name by the character set support in APR. Generally, this means that it must be supported by iconv.

### Example

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

The character set names in this example work with the iconv translation support in Solaris 8.



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_dav

[: \(WebDAV\)](#)

[: Extension](#)

[: dav\\_module](#)

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

1 2

[WebDAV \("](#)

[DavLockDB](#)

[LimitXMLRequestBody](#)

[WebDAV Resources](#)



mod\_dav httpd.conf :

```
Dav On
```

DAV DAV

DAV [DavLockDB](#) http

```
DavLockDB /usr/local/apache2/var/DavLock
```

Apache [User](#)

[<Limit>](#) [<Location>](#) DAV  
[LimitXMLRequestBody](#)

```
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

DAV  
WebDAV

HTT  
[SSL](#)

[mod\\_dav](#)  
[Group](#)

Apache

)

[mod\\_dav](#)  
[DavDepthInfinity](#)

PROPFIND

[Limi](#)

DAV



---

<u>mod_dav</u>	(PHP CGI)
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](#)
- [:](#)
- [: Extension](#)
- [: mod\\_dav](#)

## WebDAV HTTP

```
<Location /foo>  
  Dav On  
</Location>
```

On mod\_dav\_fs

filesystem

WebDAV



```
[: PROPFIND, Depth: Infinity  
[: DavDepthInfinity on|off  
[: DavDepthInfinity off  
[: , ,  
[: Extension  
[: mod_dav
```

'Depth: Infinity'            PROPFIND  
denial-of-service



```

: DAV
: DavMinTimeout seconds
: DavMinTimeout 0
: ,
: Extension
: mod_dav

```

DAV

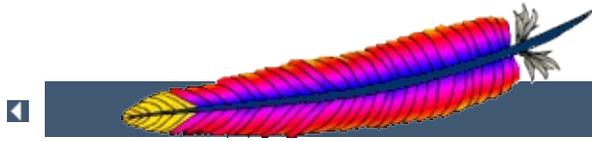
DavMin

( 600 )

```

<Location /MSWord>
  DavMinTimeout 600
</Location>

```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_dav\_fs

- [: mod\\_dav](#)
- [: Extension](#)
- [: dav\\_fs\\_module](#)
- [: mod\\_dav\\_fs.c](#)

[mod\\_dav](#)

[mod\\_dav](#)

Dav filesystem

filesystem [mod\\_dav](#)

[mod\\_dav](#)



```

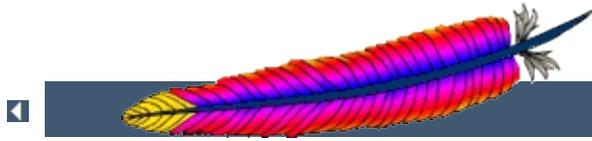
: DAV
: DavLockDB file-path
: ,
: Extension
: mod_dav_fs

```

DavLockDE

SDBM

```
DavLockDB logs/DavLock
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_deflate

- ⋮
- ⋮ Extension
- ⋮ deflate\_module
- ⋮ mod\_deflate.c

mod\_deflate DEFLATE

Filters



```
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

```
<Location />
  # Insert filter
  SetOutputFilter DEFLATE

  # Netscape 4.x has some problems...
  BrowserMatch ^Mozilla/4 gzip-only-text/html

  # Netscape 4.06-4.08 have some more problems
  BrowserMatch ^Mozilla/4\.0[678] no-gzip

  # MSIE masquerades as Netscape, but it is fine
  # BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

  # NOTE: Due to a bug in mod_setenvif up to Apache 2.0.48
  # the above regex won't work. You can use the following
  # workaround to get the desired effect:
  BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

  # Don't compress images
  SetEnvIfNoCase Request_URI \
    \.(?:gif|jpe?g|png)$ no-gzip dont-vary

  # Make sure proxies don't deliver the wrong content
  Header append Vary User-Agent env=!dont-vary
</Location>
```



---

## Output Compression

DEFLATE

```
SetOutputFilter DEFLATE
```

MIME

[AddOutputFilt](#)

```
<Directory "/your-server-root/manual">  
  AddOutputFilterByType DEFLATE text/html  
</Directory>
```

[BrowserMa](#)

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  
                                     4.06, 4.07, 4.08 html
```

3 [BrowserMatch](#)

"Mozilla/4"

U:

```
DEFLATE PHP SSI RESOURCE
```

mod\_deflate gzip

AddInputFilter

DEFLATE

```
<Location /dav-area>  
  SetInputFilter DEFLATE  
</Location>
```

Content-Encoding: gzip

**Content-Length**

*Content-Length*

Content-Length



mod\_deflate Vary: Accept-Encoding HTTP  
Accept-Encoding

User-Agent Vary  
DEFLATE

Header append Vary User-Agent

( HTTP ) Vary \*

Header set Vary \*



```
zlib
DeflateBufferSize value
DeflateBufferSize 8096
,
Extension
mod_deflate
```

DeflateBufferSize

zlib



## DeflateCompressionLevel

```
:  
: DeflateCompressionLevel value  
: Zlib  
: ,  
: Extension  
: mod_deflate  
: This directive is available since Apache 2.0.45
```

### DeflateCompressionLevel

1) 9)



```

:
: DeflateFilterNote [type] notename
: ,
: Extension
: mod_deflate
: type is available since Apache 2.0.45

```

## DeflateFilterNote

```
DeflateFilterNote ratio
```

```
LogFormat "%r" %b (%{ratio}n) "%{User-agent}i" deflate
CustomLog logs/deflate_log deflate
```

*type*

### Input

### Output

### Ratio

( / \* 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)  
)
```

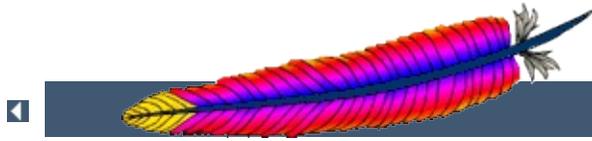


```
: Zlib
: DeflateWindowSize value
: DeflateWindowSize 15
: ,
: Extension
: mod_deflate
```

DeflateWindowSize

zlib (: zlib )

(: 2



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_dir

This translation may be out of date. Check the English version for recent changes.



:

- index.html
- mod\_dir
- 

()

http://servername/foo/dirname URL

URL

http://servername/foo/dirname/



## DirectoryIndex

```
DirectoryIndex local-url [local-url] ...  
DirectoryIndex index.html  
,, .htaccess  
Indexes  
Base  
mod_dir
```

/ URL

```
DirectoryIndex index.html
```

http://myserver/docs/  
http://myserver/docs/index.html URL

```
DirectoryIndex index.html index.txt /cgi-bin/index.pl
```

index.html index.txt CGI



## DirectorySlash

```
:  
: DirectorySlash On|Off  
: DirectorySlash On  
: , , , .htaccess  
: Indexes  
: Base  
: mod_dir  
: 2.0.51
```

URL

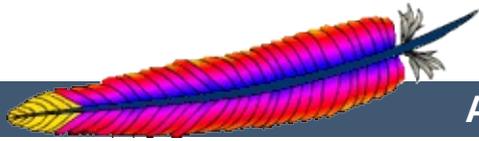
mod\_dir

- URL
- mod\_autoindex    mod\_autoindex
- DirectoryIndex
- HTML URL

:

```
# see security warning below!  
<Location /some/path>  
  DirectorySlash Off  
  SetHandler some-handler  
</Location>
```

```
DirectoryIndex (                    index.html) URL  
  index.html
```



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



## Apache HTTP Server Version 2.0

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

# Apache Module `mod_disk_cache`

<b>Description:</b>	Content cache storage manager keyed to URIs
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>disk_cache_module</code>
<b>Source File:</b>	<code>mod_disk_cache.c</code>

## Summary

This module is experimental. Documentation is still under development...

`mod_disk_cache` implements a disk based storage manager. It is primarily of use in conjunction with `mod_proxy`.

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.

### Note:

`mod_disk_cache` requires the services of `mod_cache`.



## CacheDirLength Directive

<b>Description:</b>	The number of characters in subdirectory names
<b>Syntax:</b>	CacheDirLength <i>length</i>
<b>Default:</b>	CacheDirLength 2
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

The `CacheDirLength` directive sets the number of characters for each subdirectory name in the cache hierarchy.

The result of `CacheDirLevels`\* `CacheDirLength` must not be higher than 20.

```
CacheDirLength 4
```



**Description:** The number of levels of subdirectories in the cache.

**Syntax:** `CacheDirLevels levels`

**Default:** `CacheDirLevels 3`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_disk_cache`

The `CacheDirLevels` directive sets the number of subdirectory levels in the cache. Cached data will be saved this many directory levels below the `CacheRoot` directory.

The result of `CacheDirLevels* CacheDirLength` must not be higher than 20.

```
CacheDirLevels 5
```



## CacheExpiryCheck Directive

<b>Description:</b>	Indicates if the cache observes Expires dates when seeking files
<b>Syntax:</b>	CacheExpiryCheck On Off
<b>Default:</b>	CacheExpiryCheck On
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

```
CacheExpiryCheck Off
```

The `CacheExpiryCheck` directive is currently *not* implemented.



**Description:** The time to retain unchanged cached files that match a URL

**Syntax:** `CacheGcClean hours url-string`

**Default:** `CacheGcClean ?`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_disk_cache`

More detail will be added here, when the function is implemented.

```
CacheGcClean 12 /daily_scripts
```

The `CacheGcClean` directive is currently *not* implemented.



## CacheGcDaily Directive

<b>Description:</b>	The recurring time each day for garbage collection to be run. (24 hour clock)
<b>Syntax:</b>	CacheGcDaily <i>time</i>
<b>Default:</b>	CacheGcDaily ?
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

```
CacheGcDaily 23:59
```

The `CacheGcDaily` directive is currently *not* implemented.



<b>Description:</b>	The interval between garbage collection attempts.
<b>Syntax:</b>	<code>CacheGcInterval</code> <i>hours</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_disk_cache</code>

The `CacheGcInterval` directive specifies the number of hours to wait between attempts to free up disk space.

More detail will be added here, when the function is implemented.

```
CacheGcInterval 24
```

The `CacheGcInterval` directive is currently *not* implemented.



## CacheGcMemUsage Directive

<b>Description:</b>	The maximum kilobytes of memory used for garbage collection
<b>Syntax:</b>	CacheGcMemUsage <i>KBytes</i>
<b>Default:</b>	CacheGcMemUsage ?
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

CacheGcMemUsage 16

The `CacheGcMemUsage` directive is currently *not* implemented.



**Description:** The time to retain unreferenced cached files that match a URL.

**Syntax:** `CacheGcUnused hours url-string`

**Default:** `CacheGcUnused ?`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_disk_cache`

More detail will be added here, when the function is implemented.

```
CacheGcUnused 12 /local_images
```

The `CacheGcUnused` directive is currently *not* implemented.



<b>Description:</b>	The maximum size (in bytes) of a document to be placed in the cache
<b>Syntax:</b>	CacheMaxFileSize <i>bytes</i>
<b>Default:</b>	CacheMaxFileSize 1000000
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

The `CacheMaxFileSize` directive sets the maximum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMaxFileSize 64000
```



**Description:** The minimum size (in bytes) of a document to be placed in the cache

**Syntax:** CacheMinFileSize *bytes*

**Default:** CacheMinFileSize 1

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_disk\_cache

The `CacheMinFileSize` directive sets the minimum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMinFileSize 64
```



**Description:** The directory root under which cache files are stored

**Syntax:** CacheRoot *directory*

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_disk\_cache

The `CacheRoot` directive defines the name of the directory on the disk to contain cache files. If the `mod_disk_cache` module has been loaded or compiled in to the Apache server, this directive *must* be defined. Failing to provide a value for `CacheRoot` will result in a configuration file processing error. The `CacheDirLevels` and `CacheDirLength` directives define the structure of the directories under the specified root directory.

```
CacheRoot c:/cacheroot
```



**Description:** The maximum amount of disk space that will be used by the cache in KBytes

**Syntax:** CacheSize *KBytes*

**Default:** CacheSize 1000000

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_disk\_cache

The `CacheSize` directive sets the desired disk space usage of the cache, in KBytes (1024-byte units). This directive does not put a hard limit on the size of the cache. The garbage collector will delete files until the usage is at or below the settings. Always use a value that is lower than the available disk space.

```
CacheSize 5000000
```



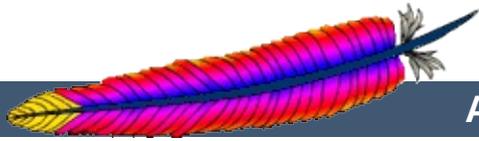
## CacheTimeMargin Directive

<b>Description:</b>	The minimum time margin to cache a document
<b>Syntax:</b>	CacheTimeMargin ?
<b>Default:</b>	CacheTimeMargin ?
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

CacheTimeMargin X

The `CacheTimeMargin` directive is currently *not* implemented.



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



## Apache HTTP Server Version 2.0

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

## Apache Module mod\_dumpio

<b>Description:</b>	Dumps all I/O to error log as desired.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	dumpio_module
<b>Source File:</b>	mod_dumpio.c

### Summary

mod\_dumpio allows for the logging of all input received by Apache and/or all output sent by Apache to be logged (dumped) to the error.log file.

The data logging is done right after SSL decoding (for input) and right before SSL encoding (for output). As can be expected, this can produce extreme volumes of data, and should only be used when debugging problems.



## Enabling sample support

To enable the module, it should be compiled and loaded in to your running Apache configuration. Logging can then be enabled or disabled via the below directives.

In order for dumping to work [LogLevel](#) must be set to debug.



## DumpIOInput Directive

<b>Description:</b>	Dump all input data to the error log
<b>Syntax:</b>	DumpIOInput On Off
<b>Default:</b>	DumpIOInput Off
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOInput is only available in Apache 2.0.53 and later.

Enable dumping of all input.

```
Example  
DumpIOInput On
```



## DumpIOOutput Directive

<b>Description:</b>	Dump all output data to the error log
<b>Syntax:</b>	DumpIOOutput On Off
<b>Default:</b>	DumpIOOutput Off
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOOutput is only available in Apache 2.0.53 and later.

Enable dumping of all output.

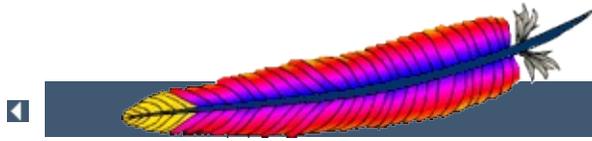
### Example

```
DumpIOOutput On
```

---

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

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



| [FAQ](#) |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

## Apache mod\_echo

This translation may be out of date. Check the English version for recent changes.

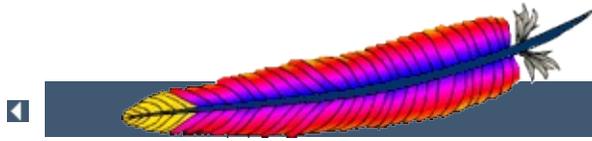
- [:](#)
- [:](#) Experimental
- [:](#) echo\_module
- [:](#) mod\_echo.c
- [:](#) Apache 2.0



```
ProtocolEcho On|Off
Experimental
mod_echo
Apache 2.0
```

## ProtocolEcho

```
ProtocolEcho On
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_env

This translation may be out of date. Check the English version for recent changes.

[CGI SSI](#)

[Base](#)

[env\\_module](#)

[mod\\_env.c](#)

CGI SSI



## PassEnv

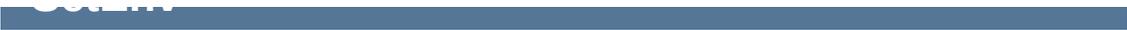
```
PassEnv env-variable [env-variable] ...  
,,, .htaccess  
FileInfo  
Base  
mod_env
```

httpd CGI

SSI

```
PassEnv LD_LIBRARY_PATH
```





```
SetEnv env-variable value  
SetEnvIf .htaccess  
FileInfo  
Base  
mod_env
```

### CGI SSI

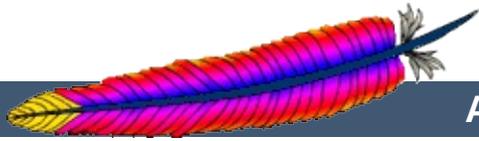
```
SetEnv SPECIAL_PATH /foo/bin
```



```
UnsetEnv env-variable [env-variable] ...  
,,, .htaccess  
FileInfo  
Base  
mod_env
```

## CGI SSI

```
UnsetEnv LD_LIBRARY_PATH
```



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



## Apache HTTP Server Version 2.0

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

# Apache Module `mod_example`

<b>Description:</b>	Illustrates the Apache module API
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>example_module</code>
<b>Source File:</b>	<code>mod_example.c</code>

## Summary

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.

The files in the `src/modules/example` directory under the Apache distribution directory tree are provided as an example to those that wish to write modules that use the Apache API.

The main file is `mod_example.c`, which illustrates all the different callback mechanisms and call syntaxes. By no means does an add-on module need to include routines for all of the callbacks - quite the contrary!

The example module is an actual working module. If you link it into your server, enable the "example-handler" handler for a location, and then browse to that location, you will see a display of some of the tracing the example module did as the various callbacks were made.



## Compiling the Example Module

To include the example module in your server, follow the steps below:

1. Uncomment the "AddModule modules/example/mod\_example" line near the bottom of the src/Configuration file. If there isn't one, add it; it should look like this:

```
AddModule modules/example/mod_example.o
```

2. Run the src/Configure script ("cd src; ./Configure"). This will build the Makefile for the server itself, and update the src/modules/Makefile for any additional modules you have requested from beneath that subdirectory.
3. Make the server (run "make" in the src directory).

To add another module of your own:

- A. `mkdir src/modules/mymodule`
- B. `cp src/modules/example/* src/modules/mymodule`
- C. Modify the files in the new directory.
- D. Follow steps [1] through [3] above, with appropriate changes.



To activate the example module, include a block similar to the following in your `srml.conf` file:

```
<Location /example-info>  
  SetHandler example-handler  
</Location>
```

As an alternative, you can put the following into a `.htaccess` file and then request the file "test.example" from that location:

```
AddHandler example-handler .example
```

After reloading/restarting your server, you should be able to browse to this location and see the brief display mentioned earlier.



## Example Directive

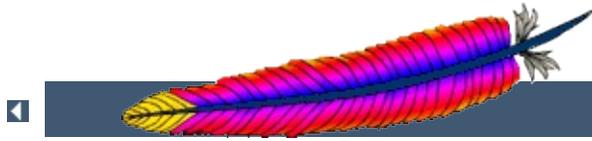
<b>Description:</b>	Demonstration directive to illustrate the Apache module API
<b>Syntax:</b>	Example
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Experimental
<b>Module:</b>	mod_example

The **Example** directive just sets a demonstration flag which the example module's content handler displays. It takes no arguments. If you browse to an URL to which the example content-handler applies, you will get a display of the routines within the module and how and in what order they were called to service the document request. The effect of this directive one can observe under the point "Example directive declared here: YES/NO".

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_expires

- [Expires](#) [Cache-Control](#)  
[HTTP](#)
- [Extension](#)
- [expires\\_module](#)
- [mod\\_expires.c](#)

[Expires HTTP](#)

[Cache-Control](#)

[max-age](#)

[max-age](#) ( [RFC 2616 section 14.9](#) )  
[Header](#)

[Cache-Control](#)



## ExpiresDefault

## ExpiresByType

:

```
ExpiresDefault "<base> [plus] {<num> <type>}"  
ExpiresByType type/encoding "<base> [plus] {<num> <type>}"
```

<base> :

- access
- now ('access')
- modification

plus <num>

( atoi() )

<t

- 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"
```

## Expires



## Expires

- Expires
- ExpiresActive On|Off
- ,,, .htaccess
- Indexes
- Extension
- mod\_expires

```
ExpiresCacheControl  
(  
    ExpiresCacheControl  
    .htaccess  
)  
  
ExpiresDefault  
ExpiresCacheControl
```



## ExpiresByType

```
: MIME Expires
: ExpiresByType MIME-type <code>seconds
: , , , .htaccess
: Indexes
: Extension
: mod_expires
```

( text/html)

## M

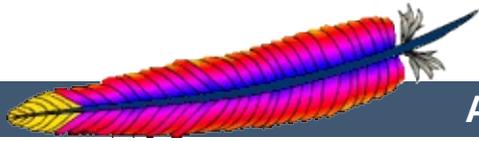
```
:
# enable expirations
ExpiresActive On
# expire GIF images after a month in the client's cache
ExpiresByType image/gif A2592000
# HTML documents are good for a week from the
# time they were changed
ExpiresByType text/html M604800
```

ExpiresActive On



## ExpiresDefault

```
:  
: ExpiresDefault <code>seconds  
: , , , .htaccess  
: Indexes  
: Extension  
: mod_expires
```



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



## Apache HTTP Server Version 2.0

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

# Apache Module `mod_ext_filter`

<b>Description:</b>	Pass the response body through an external program before delivery to the client
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>ext_filter_module</code>
<b>Source File:</b>	<code>mod_ext_filter.c</code>

## Summary

`mod_ext_filter` presents a simple and familiar programming model for [filters](#). With this module, a program which reads from stdin and writes to stdout (i.e., a Unix-style filter command) can be a filter for Apache. This filtering mechanism is much slower than using a filter which is specially written for the Apache API and runs inside of the Apache server process, but it does have the following benefits:

- the programming model is much simpler
- any programming/scripting language can be used, provided that it allows the program to read from standard input and write to standard output
- existing programs can be used unmodified as Apache filters

Even when the performance characteristics are not suitable for production use, `mod_ext_filter` can be used as a prototype environment for filters.

## See also

[Filters](#)



## Generating HTML from some other type of response

```
# mod_ext_filter directive to define a filter
# to HTML-ize text/c files using the external
# program /usr/bin/enscript, with the type of
# the result set to 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 directive to cause the new filter to
    # be run on output
    SetOutputFilter c-to-html

    # mod_mime directive to set the type of .c
    # files to text/c
    AddType text/c .c

    # mod_ext_filter directive to set the debug
    # level just high enough to see a log message
    # per request showing the configuration in force
    ExtFilterOptions DebugLevel=1
</Directory>
```

## Implementing a content encoding filter

Note: this gzip example is just for the purposes of illustration. Please refer to [mod\\_deflate](#) for a practical implementation.

```
# mod_ext_filter directive to define the external filter
ExtFilterDefine gzip mode=output cmd=/bin/gzip

<Location /gzipped>
    # core directive to cause the gzip filter to be
    # run on output
    SetOutputFilter gzip

    # mod_header directive to add
    # "Content-Encoding: gzip" header field
    Header set Content-Encoding gzip
</Location>
```

## Slowing down the server

```
# mod_ext_filter directive to define a filter
# which runs everything through cat; cat doesn't
# modify anything; it just introduces extra pathlength
# and consumes more resources
ExtFilterDefine slowdown mode=output cmd=/bin/cat \
    preservescontentlength

<Location />
    # core directive to cause the slowdown filter to
    # be run several times on output
    #
    SetOutputFilter slowdown;slowdown;slowdown
</Location>
```

## Using sed to replace text in the response

```
# mod_ext_filter directive to define a filter which
# replaces text in the response
#
ExtFilterDefine fixtext mode=output intype=text/html \
    cmd="/bin/sed s/verdana/arial/g"

<Location />
    # core directive to cause the fixtext filter to
    # be run on output
    SetOutputFilter fixtext
</Location>
```

## Tracing another filter

```
# Trace the data read and written by mod_deflate
# for a particular client (IP 192.168.1.31)
# experiencing compression problems.
# This filter will trace what goes into mod_deflate.
ExtFilterDefine tracebefore \
    cmd="/bin/tracefilter.pl /tmp/tracebefore" \
    EnableEnv=trace_this_client

# This filter will trace what goes after mod_deflate.
# Note that without the ftype parameter, the default
# filter type of AP_FTYPE_RESOURCE would cause the
# filter to be placed *before* mod_deflate in the filter
```

```
# chain. Giving it a numeric value slightly higher than
# AP_FTYPE_CONTENT_SET will ensure that it is placed
# after 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>
```

### Here is the filter which traces the data:

```
#!/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);
```



<b>Description:</b>	Define an external filter
<b>Syntax:</b>	<code>ExtFilterDefine <i>filtername</i> <i>parameters</i></code>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ext_filter</code>

The `ExtFilterDefine` directive defines the characteristics of an external filter, including the program to run and its arguments.

*filtername* specifies the name of the filter being defined. This name can then be used in `SetOutputFilter` directives. It must be unique among all registered filters. *At the present time, no error is reported by the register-filter API, so a problem with duplicate names isn't reported to the user.*

Subsequent parameters can appear in any order and define the external command to run and certain other characteristics. The only required parameter is `cmd=`. These parameters are:

#### **`cmd=cmdline`**

The `cmd=` keyword allows you to specify the external command to run. If there are arguments after the program name, the command line should be surrounded in quotation marks (e.g., `cmd="/bin/mypgm arg1 arg2"`). Normal shell quoting is not necessary since the program is run directly, bypassing the shell. Program arguments are blank-delimited. A backslash can be used to escape blanks which should be part of a program argument. Any backslashes which are part of the argument must be escaped with backslash themselves. In addition to the standard CGI environment variables, `DOCUMENT_URI`, `DOCUMENT_PATH_INFO`, and `QUERY_STRING_UNESCAPED` will also be set for the program.

**mode=*mode***

mode should be output for now (the default). In the future, mode=input will be used to specify a filter for request bodies.

**intype=*imt***

This parameter specifies the internet media type (*i.e.*, MIME type) of documents which should be filtered. By default, all documents are filtered. If intype= is specified, the filter will be disabled for documents of other types.

**outtype=*imt***

This parameter specifies the internet media type (*i.e.*, MIME type) of filtered documents. It is useful when the filter changes the internet media type as part of the filtering operation. By default, the internet media type is unchanged.

**PreservesContentLength**

The PreservesContentLength keyword specifies that the filter preserves the content length. This is not the default, as most filters change the content length. In the event that the filter doesn't modify the length, this keyword should be specified.

**ftype=*filtertype***

This parameter specifies the numeric value for filter type that the filter should be registered as. The default value, AP\_FTYPE\_RESOURCE, is sufficient in most cases. If the filter needs to operate at a different point in the filter chain than resource filters, then this parameter will be necessary. See the AP\_FTYPE\_foo definitions in util\_filter.h for appropriate values.

**disableenv=*env***

This parameter specifies the name of an environment variable which, if set, will disable the filter.

**enableenv=*env***

This parameter specifies the name of an environment variable which must be set, or the filter will be disabled.



<b>Description:</b>	Configure <code>mod_ext_filter</code> options
<b>Syntax:</b>	<code>ExtFilterOptions option [option] ...</code>
<b>Default:</b>	<code>ExtFilterOptions DebugLevel=0 NoLogStderr</code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ext_filter</code>

The `ExtFilterOptions` directive specifies special processing options for `mod_ext_filter`. *Option* can be one of

### **DebugLevel=*n***

The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_ext_filter`. By default, no debug messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the `DBGLVL_` constants near the beginning of `mod_ext_filter.c`.

Note: The core directive `LogLevel` should be used to cause debug messages to be stored in the Apache error log.

### **LogStderr | NoLogStderr**

The `LogStderr` keyword specifies that messages written to standard error by the external filter program will be saved in the Apache error log. `NoLogStderr` disables this feature.

### **Example**

```
ExtFilterOptions LogStderr DebugLevel=0
```

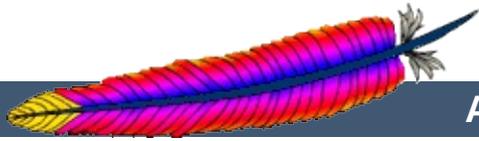
Messages written to the filter's standard error will be stored in the

Apache error log. No debug messages will be generated by [mod\\_ext\\_filter](#).

---

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_file_cache`

<b>Description:</b>	Caches a static list of files in memory
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>file_cache_module</code>
<b>Source File:</b>	<code>mod_file_cache.c</code>

## Summary

This module should be used with care. You can easily create a broken site using `mod_file_cache`, so read this document carefully.

*Caching* frequently requested files that change very infrequently is a technique for reducing server load. `mod_file_cache` provides two techniques for caching frequently requested *static* files. Through configuration directives, you can direct `mod_file_cache` to either open then `mmap()` a file, or to pre-open a file and save the file's open *file handle*. Both techniques reduce server load when processing requests for these files by doing part of the work (specifically, the file I/O) for serving the file when the server is started rather than during each request.

Notice: You cannot use this for speeding up CGI programs or other files which are served by special content handlers. It can only be used for regular files which are usually served by the Apache core content handler.

This module is an extension of and borrows heavily from the `mod_mmap_static` module in Apache 1.3.



`mod_file_cache` caches a list of statically configured files via `MMapFile` or `CacheFile` directives in the main server configuration.

Not all platforms support both directives. For example, Apache on Windows does not currently support the `MMapStatic` directive, while other platforms, like AIX, support both. You will receive an error message in the server error log if you attempt to use an unsupported directive. If given an unsupported directive, the server will start but the file will not be cached. On platforms that support both directives, you should experiment with both to see which works best for you.

## MMapFile Directive

The `MMapFile` directive of `mod_file_cache` maps a list of statically configured files into memory through the system call `mmap()`. This system call is available on most modern Unix derivatives, but not on all. There are sometimes system-specific limits on the size and number of files that can be `mmap()`ed, experimentation is probably the easiest way to find out.

This `mmap()`ing is done once at server start or restart, only. So whenever one of the mapped files changes on the filesystem you *have* to restart the server (see the [Stopping and Restarting](#) documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` and `mv` do this. The reason why this module doesn't take care of changes to the files is that this check would need an extra `stat()` every time which is a waste and against the intent of I/O reduction.

## CacheFile Directive

The [CacheFile](#) directive of [mod\\_file\\_cache](#) opens an active *handle* or *file descriptor* to the file (or files) listed in the configuration directive and places these open file handles in the cache. When the file is requested, the server retrieves the handle from the cache and passes it to the `sendfile()` (or `TransmitFile()` on Windows), socket API.

This file handle caching is done once at server start or restart, only. So whenever one of the cached files changes on the filesystem you *have* to restart the server (see the [Stopping and Restarting](#) documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` and `mv` do this.

### Note

Don't bother asking for a directive which recursively caches all the files in a directory. Try this instead... See the [Include](#) directive, and consider this command:

```
find /www/htdocs -type f -print \  
| sed -e 's/./mmapfile &/' > /www/conf/mmap.conf
```



<b>Description:</b>	Cache a list of file handles at startup time
<b>Syntax:</b>	CacheFile <i>file-path</i> [ <i>file-path</i> ] ...
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_file_cache

The **CacheFile** directive opens handles to one or more files (given as whitespace separated arguments) and places these handles into the cache at server startup time. Handles to cached files are automatically closed on a server shutdown. When the files have changed on the filesystem, the server should be restarted to re-cache them.

Be careful with the *file-path* arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links *etc.* because that again would cost extra `stat()` system calls which is not acceptable. This module may or may not work with filenames rewritten by [mod\\_alias](#) or [mod\\_rewrite](#).

### Example

```
CacheFile /usr/local/apache/htdocs/index.html
```



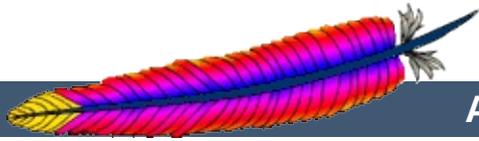
<b>Description:</b>	Map a list of files into memory at startup time
<b>Syntax:</b>	<code>MMapFile <i>file-path</i> [<i>file-path</i>] ...</code>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_file_cache</code>

The `MMapFile` directive maps one or more files (given as whitespace separated arguments) into memory at server startup time. They are automatically unmapped on a server shutdown. When the files have changed on the filesystem at least a HUP or USR1 signal should be send to the server to `re-mmap()` them.

Be careful with the *file-path* arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links *etc.* because that again would cost extra `stat()` system calls which is not acceptable. This module may or may not work with filenames rewritten by [mod\\_alias](#) or [mod\\_rewrite](#).

### Example

```
MMapFile /usr/local/apache/htdocs/index.html
```



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



## Apache HTTP Server Version 2.0

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

# Apache Module mod\_headers

<b>Description:</b>	Customization of HTTP request and response headers
<b>Status:</b>	Extension
<b>Module Identifier:</b>	headers_module
<b>Source File:</b>	mod_headers.c

## Summary

This module provides directives to control and modify HTTP request and response headers. Headers can be merged, replaced or removed.



The directives provided by `mod_headers` can occur almost anywhere within the server configuration. They are valid in the main server config and virtual host sections, inside `<Directory>`, `<Location>` and `<Files>` sections, and within `.htaccess` files.

The directives are processed in the following order:

1. main server
2. virtual host
3. `<Directory>` sections and `.htaccess`
4. `<Files>`
5. `<Location>`

Order is important. These two headers have a different effect if reversed:

```
RequestHeader append MirrorID "mirror 12"  
RequestHeader unset MirrorID
```

This way round, the `MirrorID` header is not set. If reversed, the `MirrorID` header is set to "mirror 12".



1. Copy all request headers that begin with "TS" to the response headers:

```
Header echo ^TS
```

2. Add a header, MyHeader, to the response including a timestamp for when the request was received and how long it took to begin serving the request. This header can be used by the client to intuit load on the server or in isolating bottlenecks between the client and the server.

```
Header add MyHeader "%D %t"
```

results in this header being added to the response:

```
MyHeader: D=3775428 t=991424704447256
```

3. Say hello to Joe

```
Header add MyHeader "Hello Joe. It took %D microseconds \\  
for Apache to serve this request."
```

results in this header being added to the response:

```
MyHeader: Hello Joe. It took D=3775428 microseconds for  
Apache to serve this request.
```

4. Conditionally send MyHeader on the response if and only if header "MyRequestHeader" is present on the request. This is useful for constructing headers in response to some client stimulus. Note that this example requires the services of the [mod\\_setenvif](#) module.

```
SetEnvIf MyRequestHeader value HAVE_MyRequestHeader
Header add MyHeader "%D %t mytext"
env=HAVE_MyRequestHeader
```

If the header `MyRequestHeader: value` is present on the HTTP request, the response will contain the following header:

```
MyHeader: D=3775428 t=991424704447256 mytext
```



<b>Description:</b>	Configure HTTP response headers
<b>Syntax:</b>	Header [ <i>condition</i> ] set   append   add   unset   echo <i>header</i> [ <i>value</i> ] [env=[!] <i>variable</i> ]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_headers
<b>Compatibility:</b>	<i>Condition</i> is available in version 2.0.51 and later

This directive can replace, merge or remove HTTP response headers. The header is modified just after the content handler and output filters are run, allowing outgoing headers to be modified.

The optional *condition* can be either `onsuccess` or `always`. It determines, which internal header table should be operated on. `onsuccess` stands for 2xx status codes and `always` for all status codes (including 2xx). Especially if you want to unset headers set by certain modules, you should try out, which table is affected.

The action it performs is determined by the second argument. This can be one of the following values:

### **set**

The response header is set, replacing any previous header with this name. The *value* may be a format string.

### **append**

The response header is appended to any existing header of the same name. When a new value is merged onto an existing header it is separated from the existing header with a comma. This is the HTTP standard way of giving a header

multiple values.

### **add**

The response header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name. This can lead to unforeseen consequences, and in general "append" should be used instead.

### **unset**

The response header of this name is removed, if it exists. If there are multiple headers of the same name, all will be removed.

### **echo**

Request headers with this name are echoed back in the response headers. *header* may be a regular expression.

This argument is followed by a *header* name, which can include the final colon, but it is not required. Case is ignored for set, append, add and unset. The *header* name for echo is case sensitive and may be a regular expression.

For add, append and set a *value* is specified as the third argument. If *value* contains spaces, it should be surrounded by doublequotes. *value* may be a character string, a string containing format specifiers or a combination of both. The following format specifiers are supported in *value*:

%t	The time the request was received in Universal Coordinated Time since the epoch (Jan. 1, 1970) measured in microseconds. The value is preceded by t=.
%D	The time from when the request was received to the time the headers are sent on the wire. This is a measure of the duration of the request. The value is

	preceded by D=.
% {FOOBAR}e	The contents of the <a href="#">environment variable</a> FOOBAR.

When the **Header** directive is used with the add, append, or set argument, a fourth argument may be used to specify conditions under which the action will be taken. If the [environment variable](#) specified in the env= . . . argument exists (or if the environment variable does not exist and env=! . . . is specified) then the action specified by the **Header** directive will take effect. Otherwise, the directive will have no effect on the request.

The **Header** directives are processed just before the response is sent to the network. These means that it is possible to set and/or override most headers, except for those headers added by the header filter.



## RequestHeader Directive

<b>Description:</b>	Configure HTTP request headers
<b>Syntax:</b>	<code>RequestHeader set append add unset header [value [env=[!]variable]]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_headers

This directive can replace, merge or remove HTTP request headers. The header is modified just before the content handler is run, allowing incoming headers to be modified. The action it performs is determined by the first argument. This can be one of the following values:

### **set**

The request header is set, replacing any previous header with this name

### **append**

The request header is appended to any existing header of the same name. When a new value is merged onto an existing header it is separated from the existing header with a comma. This is the HTTP standard way of giving a header multiple values.

### **add**

The request header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name. This can lead to unforeseen consequences, and in general append should be used instead.

### **unset**

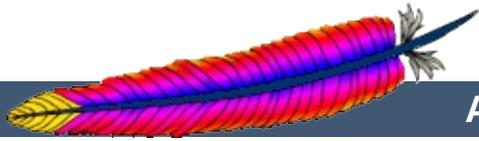
The request header of this name is removed, if it exists. If

there are multiple headers of the same name, all will be removed.

This argument is followed by a header name, which can include the final colon, but it is not required. Case is ignored. For add, append and set a *value* is given as the third argument. If *value* contains spaces, it should be surrounded by double quotes. For unset, no *value* should be given.

When the **RequestHeader** directive is used with the add, append, or set argument, a fourth argument may be used to specify conditions under which the action will be taken. If the [environment variable](#) specified in the env= . . . argument exists (or if the environment variable does not exist and env=! . . . is specified) then the action specified by the **RequestHeader** directive will take effect. Otherwise, the directive will have no effect on the request.

The **RequestHeader** directive is processed just before the request is run by its handler in the fixup phase. This should allow headers generated by the browser, or by Apache input filters to be overridden or modified.



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



## Apache HTTP Server Version 2.0

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

# Apache Module mod\_imap

<b>Description:</b>	Server-side imagemap processing
<b>Status:</b>	Base
<b>Module Identifier:</b>	imap_module
<b>Source File:</b>	mod_imap.c

## Summary

This module processes `.map` files, thereby replacing the functionality of the `imagemap` CGI program. Any directory or document type configured to use the handler `imap-file` (using either [AddHandler](#) or [SetHandler](#)) will be processed by this module.

The following directive will activate files ending with `.map` as imagemap files:

```
AddHandler imap-file map
```

Note that the following is still supported:

```
AddType application/x-httpd-imap map
```

However, we are trying to phase out "magic MIME types" so we are deprecating this method.



## NEW FEATURES

The imagemap module adds some new features that were not possible with previously distributed imagemap programs.

- URL references relative to the Referer: information.
- Default <base> assignment through a new map directive base.
- No need for `imagemap.conf` file.
- Point references.
- Configurable generation of imagemap menus.



The lines in the imagemap files can have one of several formats:

```
directive value [x,y ...]
directive value "Menu text" [x,y ...]
directive value x,y ... "Menu text"
```

The directive is one of `base`, `default`, `poly`, `circle`, `rect`, or `point`. The value is an absolute or relative URL, or one of the special values listed below. The coordinates are `x, y` pairs separated by whitespace. The quoted text is used as the text of the link if a imagemap menu is generated. Lines beginning with '#' are comments.

## Imagemap File Directives

There are six directives allowed in the imagemap file. The directives can come in any order, but are processed in the order they are found in the imagemap file.

### base Directive

Has the effect of `<base href="value">`. The non-absolute URLs of the map-file are taken relative to this value. The base directive overrides [ImapBase](#) as set in a `.htaccess` file or in the server configuration files. In the absence of an [ImapBase](#) configuration directive, base defaults to `http://server_name/`.

`base_uri` is synonymous with `base`. Note that a trailing slash on the URL is significant.

### default Directive

The action taken if the coordinates given do not fit any of the `poly`, `circle` or `rect` directives, and there are no `point` directives. Defaults to `nocontent` in the absence of an

ImapDefault configuration setting, causing a status code of 204 No Content to be returned. The client should keep the same page displayed.

### **poly Directive**

Takes three to one-hundred points, and is obeyed if the user selected coordinates fall within the polygon defined by these points.

### **circle**

Takes the center coordinates of a circle and a point on the circle. Is obeyed if the user selected point is with the circle.

### **rect Directive**

Takes the coordinates of two opposing corners of a rectangle. Obeyed if the point selected is within this rectangle.

### **point Directive**

Takes a single point. The point directive closest to the user selected point is obeyed if no other directives are satisfied. Note that default will not be followed if a point directive is present and valid coordinates are given.

## **Values**

The values for each of the directives can any of the following:

### **a URL**

The URL can be relative or absolute URL. Relative URLs can contain '..' syntax and will be resolved relative to the base value.

base itself will not resolved according to the current value. A statement base `mailto:` will work properly, though.

### **map**

Equivalent to the URL of the imagemap file itself. No

coordinates are sent with this, so a menu will be generated unless [ImapMenu](#) is set to none.

### **menu**

Synonymous with map.

### **referer**

Equivalent to the URL of the referring document. Defaults to `http://servername/` if no `Referer:` header was present.

### **nocontent**

Sends a status code of 204 No Content, telling the client to keep the same page displayed. Valid for all but base.

### **error**

Fails with a 500 Server Error. Valid for all but base, but sort of silly for anything but default.

## **Coordinates**

### **0,0 200,200**

A coordinate consists of an *x* and a *y* value separated by a comma. The coordinates are separated from each other by whitespace. To accommodate the way Lynx handles imagemaps, should a user select the coordinate 0, 0, it is as if no coordinate had been selected.

## **Quoted Text**

### **"Menu Text"**

After the value or after the coordinates, the line optionally may contain text within double quotes. This string is used as the text for the link if a menu is generated:

```
<a href="http://foo.com/">Menu text</a>
```

If no quoted text is present, the name of the link will be used

as the text:

```
<a href="http://foo.com/">http://foo.com</a>
```

If you want to use double quotes within this text, you have to write them as `&quot;` ; .



## Example menu

```
#Comments are printed in a 'formatted' or 'semiformatted' menu.  
#And can contain html tags. <hr>  
base referer  
poly map "Could I have a menu, please?" 0,0 0,10 10,10 10,0  
rect .. 0,0 77,27 "the directory of the referer"  
circle http://www.inetnebr.com/lincoln/feedback/ 195,0 305,27  
rect another_file "in same directory as referer" 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 "Bugs?"
```



### HTML example

```
<a href="/maps/imagemap1.map">  
    
</a>
```

### XHTML example

```
<a href="/maps/imagemap1.map">  
    
</a>
```



<b>Description:</b>	Default base for imagemap files
<b>Syntax:</b>	ImapBase map referrer URL
<b>Default:</b>	ImapBase http://servername/
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_imap

The **ImapBase** directive sets the default base used in the imagemap files. Its value is overridden by a base directive within the imagemap file. If not present, the base defaults to `http://servername/`.

### See also

- [UseCanonicalName](#)



## imapDefault

<b>Description:</b>	Default action when an imagemap is called with coordinates that are not explicitly mapped
<b>Syntax:</b>	ImapDefault error   nocontent   map   referer   <i>URL</i>
<b>Default:</b>	ImapDefault nocontent
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_imap

The `ImapDefault` directive sets the default default used in the imagemap files. Its value is overridden by a default directive within the imagemap file. If not present, the default action is nocontent, which means that a 204 No Content is sent to the client. In this case, the client should continue to display the original page.



## IMPLEMENTATION DETAILS

<b>Description:</b>	Action if no coordinates are given when calling an imagemap
<b>Syntax:</b>	ImapMenu none   formatted   semiformatted   unformatted
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_imap

The **ImapMenu** directive determines the action taken if an imagemap file is called without valid coordinates.

### **none**

If ImapMenu is none, no menu is generated, and the default action is performed.

### **formatted**

A formatted menu is the simplest menu. Comments in the imagemap file are ignored. A level one header is printed, then an hrule, then the links each on a separate line. The menu has a consistent, plain look close to that of a directory listing.

### **semiformatted**

In the semiformatted menu, comments are printed where they occur in the imagemap file. Blank lines are turned into HTML breaks. No header or hrule is printed, but otherwise the menu is the same as a formatted menu.

### **unformatted**

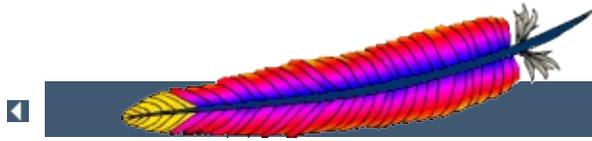
Comments are printed, blank lines are ignored. Nothing is printed that does not appear in the imagemap file. All breaks and headers must be included as comments in the imagemap file. This gives you the most flexibility over the appearance of your menus, but requires you to treat your map files as HTML

instead of plaintext.

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_include

This translation may be out of date. Check the English version for recent changes.

- [html \(Server Side Includes\)](#)
- [Base](#)
- [include\\_module](#)
- [mod\\_include.c](#)
- [Apache 2.0](#)

[Options](#)

[AcceptPathInfo](#)

[SSI](#)



Server Side Includes INCLUDES Server-side  
.shtml Apache

```
AddType text/html .shtml  
AddOutputFilter INCLUDES .shtml
```

shtml ( Options .htaccess ):

```
Options +Includes
```

```
server-parsed INCLUDES MIME  
server-parsed-html text/x-server-parsed-html3  
Apache INCLUDES ( MIME
```

[Tutorial on Server Side Includes.](#)



SSl PATH\_INFO ()



SGML

HTML

```
<!--#element attribute=value attribute=value ... -->
```

(: value) ( ) ( )  
( -->) SSI

(: element)

<a href="#">config</a>	configure output formats
<a href="#">echo</a>	print variables
<a href="#">exec</a>	execute external programs
<a href="#">fsize</a>	print size of a file
<a href="#">flastmod</a>	print last modification time of a file
<a href="#">include</a>	include a file
<a href="#">printenv</a>	print all available variables
<a href="#">set</a>	set a value of a variable

SSI [mod\\_include](#)

[exe](#)

**config**

**errmsg**

**sizeofmt**

)

**timefmt**

strftime(3)

## echo

[include](#) [SSIUndefinedEcho](#)

## var

## encoding

```
        echo                entity
encoding                encoding
encoding var                ISO-8859-1
```



## exec

exec CGI [mod](#)

## cgi

```
(%-) URL                (/)
(ScriptAlias Option ExecCGI ) CGI
CGI                PATH_INFO (
CGI include
```



```
<!--#exec cgi="/cgi-bin/example.cgi" -->
```

Location: HTML ()

exec cgi [include virtual](#) CGI

cgi                    include virtual

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value"
-->
```

## cmd

/bin/sh            CGI                    [inclu](#)

```
  #include virtual #exec cgi #exec cmd
(        #include virtual) Apache
```

Win32            [suexec](#) unix                    exec  
unix suexec                    Win32 suexec unix :

```
<!--#exec cmd="perl /path/to/perlscript arg1 arg2" -->
```

## fsize

  sizeofmt

## file

## virtual

(%) URL-path (/)

## flastmod

  timefmt

## include

(text/plain, text/html)

include

## file

.../

## virtual

(%) URL

URL

URL URL

URL CGI

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value" -->
```

HTML CGI

includ

## printenv

Apache 1.3.12

(

```
<!--#printenv -->
```

## set

## var

## value

```
<!--#set var="category" value="help" -->
```



---

CGI    echo            if    elif,

**DATE\_GMT**

**DATE\_LOCAL**

**DOCUMENT\_NAME**

()

**DOCUMENT\_URI**

(% ) URL-path

**LAST\_MODIFIED**

**QUERY\_STRING\_UNESCAPED**

(%-)

shell



## SSI

echo, set

:

```
<!--#if expr="$a = \$test" -->
```

:

```
<!--#set var="Zed" value="{REMOTE_HOST}_{REQUEST_METHOD}" -->
```

```
REMOTE_HOST "X" REQUEST_METHOD "Y" Zed  
"X_Y"
```

```
DOCUMENT_URI /foo/file.html "in foo"  
/bar/file.html "in bar" "in neither"
```

```
<!--#if expr='"$DOCUMENT_URI" = "/foo/file.html"' -->  
in foo  
<!--#elif expr='"$DOCUMENT_URI" = "/bar/file.html"' -->  
in bar  
<!--#else -->  
in neither  
<!--#endif -->
```



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

```
if if
elif else test_condition
endif if
```

*test\_condition :*

**string**

*string*

**string1 = string2**

**string1 == string2**

**string1 != string2**

*string1 string2*

*string2 /string/*

[perl 5](#)

**== =**

( = ==)

```
<!--#if expr="$QUERY_STRING = /^sid=[a-zA-Z0-9+]/" -->
<!--#set var="session" value="$1" -->
<!--#endif -->
```

**string1 < string2**

**string1 <= string2**

**string1 > string2**

**string1 >= string2**

*string1 string2*

( strcmp(3) )

"100"

"20"

**( test\_condition )**

test\_condition

**! test\_condition**

test\_condition

**test\_condition1 && test\_condition2**

test\_condition1 test\_condition2

**test\_condition1 || test\_condition2**

test\_condition1 test\_condition2

"=" " !=" " &&" " !" :

```
<!--#if expr="$a = test1 && $b = test2" -->  
<!--#if expr="($a = test1) && ($b = test2)" -->
```

&& ||

:

```
string1 string2 string1 string2
```

```
'string1 string2' string1 string2
```



```
include  
SSIEndTag tag  
SSIEndTag "-->"  
,  
Base  
mod_include  
2.0.30
```

mod\_include include

```
SSIEndTag "%>"
```

- SSIStartTag



- [: SSI](#)
- [: SSIErrorMsg \*message\*](#)
- [: SSIErrorMsg "\[an error occurred while processing this directive\]"](#)
- [: , , , .htaccess](#)
- [: All](#)
- [: Base](#)
- [: mod\\_include](#)
- [: 2.0.30](#)

SSLErrorMsg [mod\\_include](#)

```
<!--#config errmsg=message -->
```

```
SSLErrorMsg "<!-- Error -->"
```



## SSICurrentTag

```
: include
: SSISartTag tag
: SSISartTag "<!--#"
: ,
: Base
: mod_include
: 2.0.30
```

mod\_include include

()

```
SSISartTag "<%"
SSISendTag "%>"
```

SSISendTag

SSI :

```
SSI
<%printenv %>
```

- SSISendTag



```
⋮  
⋮ SSITimeFormat formatstring  
⋮ SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"  
⋮ , , , .htaccess  
⋮ All  
⋮ Base  
⋮ mod_include  
⋮ 2.0.30
```

DATE echo

```
<!--#config timefmt=formatstring -->
```

```
SSITimeFormat "%R, %B %d, %Y"
```

"22:26, June 14, 2002"



```
⋮ echo
⋮ SSIUndefinedEcho string
⋮ SSIUndefinedEcho "(none)"
⋮ ,
⋮ All
⋮ Base
⋮ mod_include
⋮ 2.0.34
```

"echo"

mod

```
SSIUndefinedEcho "<!-- undef -->"
```



```

: SSI
: XBitHack on|off|full
: XBitHack off
: , , .htaccess
: Options
: Base
: mod_include

```

XBitHack HTML

MIME

**off**

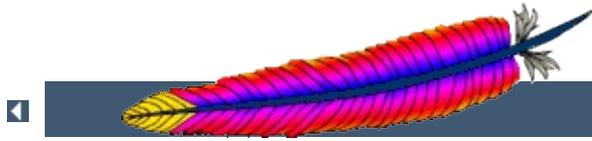
**on**

text/html html

**full**

on

```
CGI #include
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_info

This translation may be out of date. Check the English version for recent changes.

- Extension
- info\_module
- mod\_info.c

mod\_info httpd.conf

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

<Location>

http://your.host.dom/server-info

mod\_info

( .ht

/

Apache

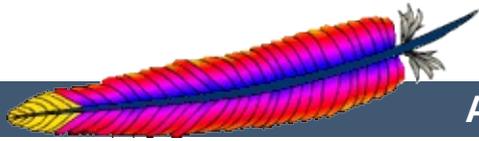


## APACHE 2.0

```
: server-info
: AddModuleInfo module-name string
: ,
: Extension
: mod_info
: Apache 1.3
```

*string module-name* HTML :

```
AddModuleInfo mod_authn_file.c 'See <a \
href="http://www.apache.org/docs/2.0/mod/mod_authn_file.html">
http://www.apache.org/docs/2.0/mod/mod_authn_file.html</a>'
```



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



## Apache HTTP Server Version 2.0

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

## Apache Module mod\_isapi

<b>Description:</b>	ISAPI Extensions within Apache for Windows
<b>Status:</b>	Base
<b>Module Identifier:</b>	isapi_module
<b>Source File:</b>	mod_isapi.c
<b>Compatibility:</b>	Win32 only

### Summary

This module implements the Internet Server extension API. It allows Internet Server extensions (e.g. ISAPI .dll modules) to be served by Apache for Windows, subject to the noted restrictions.

ISAPI extension modules (.dll files) are written by third parties. The Apache Group does not author these modules, so we provide no support for them. Please contact the ISAPI's author directly if you are experiencing problems running their ISAPI extension. **Please do not post such problems to Apache's lists or bug reporting pages.**



In the server configuration file, use the [AddHandler](#) directive to associate ISAPI files with the `isapi-handler` handler, and map it to them with their file extensions. To enable any `.dll` file to be processed as an ISAPI extension, edit the `httpd.conf` file and add the following line:

```
AddHandler isapi-handler .dll
```

In versions of the Apache server prior to 2.0.37, use `isapi-isa` instead of `isapi-handler`. The new handler name is not available prior to version 2.0.37. For compatibility, configurations may continue using `isapi-isa` through all versions of Apache prior to 2.3.0.

There is no capability within the Apache server to leave a requested module loaded. However, you may preload and keep a specific module loaded by using the following syntax in your `httpd.conf`:

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

Whether or not you have preloaded an ISAPI extension, all ISAPI extensions are governed by the same permissions and restrictions as CGI scripts. That is, [Options](#) `ExecCGI` must be set for the directory that contains the ISAPI `.dll` file.

Review the [Additional Notes](#) and the [Programmer's Journal](#) for additional details and clarification of the specific ISAPI support offered by [mod\\_isapi](#).



---

Apache's ISAPI implementation conforms to all of the ISAPI 2.0 specification, except for some "Microsoft-specific" extensions dealing with asynchronous I/O. Apache's I/O model does not allow asynchronous reading and writing in a manner that the ISAPI could access. If an ISA tries to access unsupported features, including async I/O, a message is placed in the error log to help with debugging. Since these messages can become a flood, the directive `ISAPILogNotSupported Off` exists to quiet this noise.

Some servers, like Microsoft IIS, load the ISAPI extension into the server and keep it loaded until memory usage is too high, or unless configuration options are specified. Apache currently loads and unloads the ISAPI extension each time it is requested, unless the `ISAPICacheFile` directive is specified. This is inefficient, but Apache's memory model makes this the most effective method. Many ISAPI modules are subtly incompatible with the Apache server, and unloading these modules helps to ensure the stability of the server.

Also, remember that while Apache supports ISAPI Extensions, it **does not support ISAPI Filters**. Support for filters may be added at a later date, but no support is planned at this time.



If you are programming Apache 2.0 `mod_isapi` modules, you must limit your calls to `ServerSupportFunction` to the following directives:

#### **HSE\_REQ\_SEND\_URL\_REDIRECT\_RESP**

Redirect the user to another location.  
This must be a fully qualified URL (e.g. `http://server/location`).

#### **HSE\_REQ\_SEND\_URL**

Redirect the user to another location.  
This cannot be a fully qualified URL, you are not allowed to pass the protocol or a server name (e.g. simply `/location`).  
This redirection is handled by the server, not the browser.

#### **Warning**

In their recent documentation, Microsoft appears to have abandoned the distinction between the two `HSE_REQ_SEND_URL` functions. Apache continues to treat them as two distinct functions with different requirements and behaviors.

#### **HSE\_REQ\_SEND\_RESPONSE\_HEADER**

Apache accepts a response body following the header if it follows the blank line (two consecutive newlines) in the headers string argument. This body cannot contain NULLs, since the headers argument is NULL terminated.

#### **HSE\_REQ\_DONE\_WITH\_SESSION**

Apache considers this a no-op, since the session will be finished when the ISAPI returns from processing.

#### **HSE\_REQ\_MAP\_URL\_TO\_PATH**

Apache will translate a virtual name to a physical name.

## **HSE\_APPEND\_LOG\_PARAMETER**

This logged message may be captured in any of the following logs:

- in the `\"%{isapi-parameter}n\"` component in a [CustomLog](#) directive
- in the `%q` log component with the [ISAPIAppendLogToQuery On](#) directive
- in the error log with the [ISAPIAppendLogToErrors On](#) directive

The first option, the `%{isapi-parameter}n` component, is always available and preferred.

## **HSE\_REQ\_IS\_KEEP\_CONN**

Will return the negotiated Keep-Alive status.

## **HSE\_REQ\_SEND\_RESPONSE\_HEADER\_EX**

Will behave as documented, although the `fKeepConn` flag is ignored.

## **HSE\_REQ\_IS\_CONNECTED**

Will report false if the request has been aborted.

Apache returns FALSE to any unsupported call to `ServerSupportFunction`, and sets the `GetLastError` value to `ERROR_INVALID_PARAMETER`.

`ReadClient` retrieves the request body exceeding the initial buffer (defined by [ISAPIReadAheadBuffer](#)). Based on the [ISAPIReadAheadBuffer](#) setting (number of bytes to buffer prior to calling the ISAPI handler) shorter requests are sent complete to the extension when it is invoked. If the request is longer, the ISAPI extension must use `ReadClient` to retrieve the remaining request body.

`WriteClient` is supported, but only with the `HSE_IO_SYNC` flag or no option flag (value of 0). Any other `WriteClient` request will be rejected with a return value of `FALSE`, and a `GetLastError` value of `ERROR_INVALID_PARAMETER`.

`GetServerVariable` is supported, although extended server variables do not exist (as defined by other servers.) All the usual Apache CGI environment variables are available from `GetServerVariable`, as well as the `ALL_HTTP` and `ALL_RAW` values.

Apache 2.0 [`mod\_isapi`](#) supports additional features introduced in later versions of the ISAPI specification, as well as limited emulation of async I/O and the `TransmitFile` semantics. Apache also supports preloading ISAPI .dlls for performance, neither of which were not available under Apache 1.3 `mod_isapi`.



<b>Description:</b>	Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log
<b>Syntax:</b>	ISAPIAppendLogToErrors on off
<b>Default:</b>	ISAPIAppendLogToErrors off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Record HSE\_APPEND\_LOG\_PARAMETER requests from ISAPI extensions to the server error log.



## ISAPIAppendLogToQuery Directive

<b>Description:</b>	Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field
<b>Syntax:</b>	ISAPIAppendLogToQuery on off
<b>Default:</b>	ISAPIAppendLogToQuery on
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Record HSE\_APPEND\_LOG\_PARAMETER requests from ISAPI extensions to the query field (appended to the [CustomLog](#) %q component).



<b>Description:</b>	ISAPI .dll files to be loaded at startup
<b>Syntax:</b>	ISAPICacheFile <i>file-path</i> [ <i>file-path</i> ] ...
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Specifies a space-separated list of file names to be loaded when the Apache server is launched, and remain loaded until the server is shut down. This directive may be repeated for every ISAPI .dll file desired. The full path name of each file should be specified. If the path name is not absolute, it will be treated relative to [ServerRoot](#).



<b>Description:</b>	Fake asynchronous support for ISAPI callbacks
<b>Syntax:</b>	ISAPIFakeAsync on off
<b>Default:</b>	ISAPIFakeAsync off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

While set to on, asynchronous support for ISAPI callbacks is simulated.



<b>Description:</b>	Log unsupported feature requests from ISAPI extensions
<b>Syntax:</b>	ISAPILogNotSupported on off
<b>Default:</b>	ISAPILogNotSupported off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Logs all requests for unsupported features from ISAPI extensions in the server error log. This may help administrators to track down problems. Once set to on and all desired ISAPI modules are functioning, it should be set back to off.



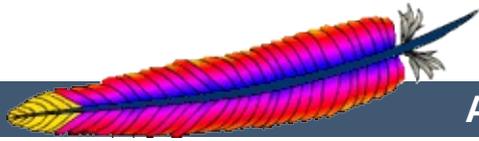
<b>Description:</b>	Size of the Read Ahead Buffer sent to ISAPI extensions
<b>Syntax:</b>	ISAPIReadAheadBuffer <i>size</i>
<b>Default:</b>	ISAPIReadAheadBuffer 49152
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Defines the maximum size of the Read Ahead Buffer sent to ISAPI extensions when they are initially invoked. All remaining data must be retrieved using the `ReadClient` callback; some ISAPI extensions may not support the `ReadClient` function. Refer questions to the ISAPI extension's author.

---

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\_ldap

<b>Description:</b>	LDAP connection pooling and result caching services for use by other LDAP modules
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	ldap_module
<b>Source File:</b>	util_ldap.c
<b>Compatibility:</b>	Available in version 2.0.41 and later

### Summary

This module was created to improve the performance of websites relying on backend connections to LDAP servers. In addition to the functions provided by the standard LDAP libraries, this module adds an LDAP connection pool and an LDAP shared memory cache.

To enable this module, LDAP support must be compiled into apr-util. This is achieved by adding the `--with-ldap` flag to the [configure](#) script when building Apache.

SSL 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.



<b>Description:</b>	Maximum number of entries in the primary LDAP cache
<b>Syntax:</b>	LDAPCacheEntries <i>number</i>
<b>Default:</b>	LDAPCacheEntries 1024
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the maximum size of the primary LDAP cache. This cache contains successful search/binds. Set it to 0 to turn off search/bind caching. The default size is 1024 cached searches.



<b>Description:</b>	Time that cached items remain valid
<b>Syntax:</b>	LDAPCacheTTL <i>seconds</i>
<b>Default:</b>	LDAPCacheTTL 600
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the time (in seconds) that an item in the search/bind cache remains valid. The default is 600 seconds (10 minutes).



<b>Description:</b>	Specifies the socket connection timeout in seconds
<b>Syntax:</b>	LDAPConnectionTimeout <i>seconds</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Time that entries in the operation cache remain valid
<b>Syntax:</b>	LDAPOpCacheTTL <i>seconds</i>
<b>Default:</b>	LDAPOpCacheTTL 600
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the time (in seconds) that entries in the operation cache remain valid. The default is 600 seconds.



<b>Description:</b>	Sets the shared memory cache file
<b>Syntax:</b>	LDAPSharedCacheFile <i>directory-path/filename</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the directory path and file name of the shared memory cache file. If not set, anonymous shared memory will be used if the platform supports it.



<b>Description:</b>	Size in bytes of the shared-memory cache
<b>Syntax:</b>	LDAPSharedCacheSize <i>bytes</i>
<b>Default:</b>	LDAPSharedCacheSize 102400
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	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.



<b>Description:</b>	Sets the file containing the trusted Certificate Authority certificate or database
<b>Syntax:</b>	LDAPTrustedCA <i>directory-path/filename</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	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`.



<b>Description:</b>	Specifies the type of the Certificate Authority file
<b>Syntax:</b>	LDAPTrustedCAType <i>type</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

The following types are supported:

DER\_FILE - file in binary DER format

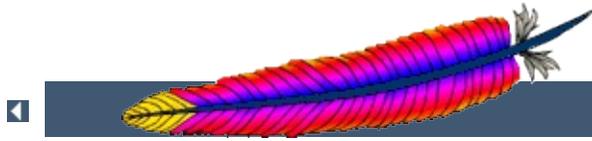
BASE64\_FILE - file in Base64 format

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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_log\_config

This translation may be out of date. Check the English version for recent changes.

- [:](#)
- [:](#) Base
- [:](#) log\_config\_module
- [:](#) mod\_log\_config.c

[:](#)  
TransferLog

[:](#)  
CustomLog

[Tran](#)

[Apache](#)



## LogFormat

## CustomLog

" %" "

"%" "

%%	( Apache 2.0.44 )
%. . . a	IP
%. . . A	IP
%. . . B	HTTP
%. . . b	HTTP CLF 1 0
%. . . {Foobar}C	<i>Foobar</i>
%. . . D	
%. . . {FOOBAR}e	<i>FOOBAR</i>
%. . . f	
%. . . h	
%. . . H	
%. . . {Foobar}i	<i>Foobar :</i>
%. . . l	(identd ) <a href="#">Ide</a> -
%. . . m	
%. . . {Foobar}n	<i>Foobar</i>
%. . . {Foobar}o	<i>Foobar :</i>
%. . . p	
%. . . P	ID
%. . .	

{format}P	ID ID 2.0.46 )	form
%...q	( ? )	
%...r		
%...s		---
%...t	CLF ()	
%... {format}t	format format	strftime
%...T		
%...u	(( %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: 5px 0;">X = + = - =</div> ( Apache 1.3 {var}c )	
%...I		0
%...O	0	

"..." ( "%h %u %r %s %b" ) ( ( "%400,501{User-agent}i" )  
"!"  
Request Not Implemented) User-agent :  
"%!200,304,302{Referer}i" Refe  
"<" ">"

%>s

httpd 2.0 1.3.25 %...r,%...i,%...o  
Log Format

2.0.46  
C ( \n, \t )

:

### **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 ()**

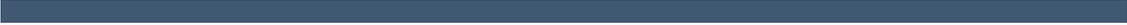
"%{User-agent}i"

%v %p

ServerName

UseCanonicalName





- [\[:\]](#) 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

```
CookieLog filename  
,  
Base  
mod_log_config
```

CookieLog

filename



## CustomLog

```
CustomLog file|pipe format|nickname [env=  
[!]environment-variable]  
,  
Base  
mod_log_config
```

### CustomLog

:

#### **file**

ServerRoot

#### **pipe**

" |"

httpd

Unix

### LogFormat

:

```
# CustomLog with format nickname  
LogFormat "%h %l %u %t \"%r\" %>s %b" common  
CustomLog logs/access_log common  
  
# CustomLog with explicit format string  
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b"
```

mod\_setenvif

mod\_rewrite

```
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  
LogFormat *nickname*  
LogFormat *format* *nickname*  
LogFormat CustomLog

*nickname* Nickname (

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost_common
```

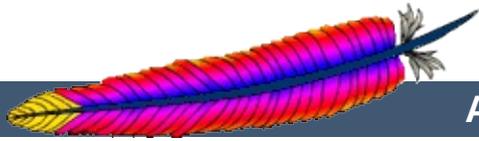


## TransferLog

```
TransferLog file|pipe
,
Base
mod_log_config
```

### Log Format

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
TransferLog 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`

<b>Description:</b>	Forensic Logging of the requests made to the server
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>log_forensic_module</code>
<b>Source File:</b>	<code>mod_log_forensic.c</code>
<b>Compatibility:</b>	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

<b>Description:</b>	Sets filename of the forensic log
<b>Syntax:</b>	ForensicLog <i>filename pipe</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_log_forensic

The **ForensicLog** directive is used to log requests to the server for forensic analysis. Each log entry is assigned a unique ID which can be associated with the request using the normal **CustomLog** directive. **mod\_log\_forensic** 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.

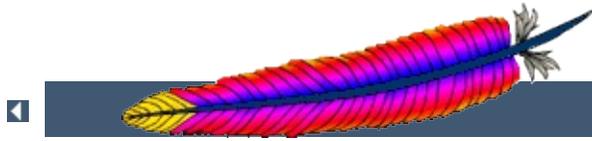
### **Note**

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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_logio

- ⋮
- ⋮ Extension
- ⋮ logio\_module
- ⋮ mod\_logio.c

mod\_log\_config

mod\_log\_config  
Apache



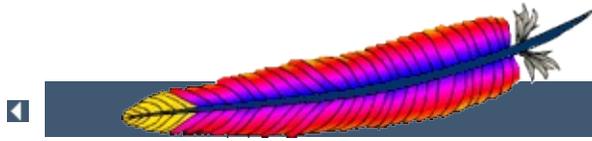
%...I	
%...0	0

0

:

**I/O :**

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"  
\"%{User-agent}i\" %I %0"
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_mem\_cache

- [URI](#)
- [Experimental](#)
- [mem\\_cache\\_module](#)
- [mod\\_mem\\_cache.c](#)

...

[mod\\_cache](#)      [mod\\_cache](#)  
[mod\\_proxy](#)    [ProxyPass](#) ( )

URI

[mod\\_cache](#)  
[mod\\_disk\\_cache](#)



## MCacheMaxObjectCount

```
┆  
┆ MCacheMaxObjectCount value  
┆ MCacheMaxObjectCount 1009  
┆  
┆ Experimental  
┆ mod_mem_cache
```

MCacheMaxObjectCount  
MCacheRemovalAlgorithm

```
MCacheMaxObjectCount 13001
```



## MCACHE\_MAXOBJECTSIZE

```
[: ] ()  
[: ] MCacheMaxObjectSize bytes  
[: ] MCacheMaxObjectSize 10000  
[: ]  
[: ] Experimental  
[: ] mod_mem_cache
```

### MCacheMaxObjectSize

```
MCacheMaxObjectSize 6400000
```

```
MCacheMaxObjectSize MCacheMinObjectSize
```



## MCacheMaxStreamingBuffer

```
[:  
[: MCacheMaxStreamingBuffer size_in_bytes  
[: MCacheMaxStreamingBuffer of 100000  
  MCacheMaxObjectSize  
[:  
[: Experimental  
[: mod_mem_cache
```

MCacheMaxStreamingBuffer

Length

MCacheMaxStreamingBuffer

Content-Ler

```
:  
MCacheMaxStreamingBuffer
```

```
# Enable caching of streamed responses up to 64KB:  
MCacheMaxStreamingBuffer 65536
```



## MCacheMinObjectSize

```
⋮ ()  
⋮ MCacheMinObjectSize bytes  
⋮ MCacheMinObjectSize 0  
⋮  
⋮ Experimental  
⋮ mod_mem_cache
```

### MCacheMinObjectSize

```
MCacheMinObjectSize 10000
```



- ┆
- ┆ MCacheRemovalAlgorithm LRU|GDSF
- ┆ MCacheRemovalAlgorithm GDSF
- ┆
- ┆ Experimental
- ┆ mod\_mem\_cache

MCacheRemovalAlgorithm

**LRU (Least Recently Used)**

LRU

**GDSF (GreedyDual-Size)**

GDSF

```
MCacheRemovalAlgorithm GDSF  
MCacheRemovalAlgorithm LRU
```

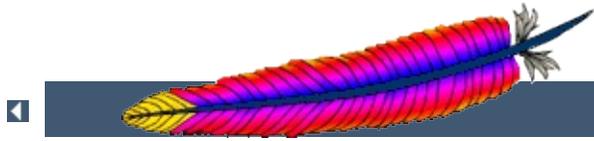


```
┆
┆ MCacheSize KBytes
┆ MCacheSize 100
┆
┆ Experimental
┆ mod_mem_cache
```

**MCacheSize** (1024 )  
MCacheRemovalAlgorithm

```
MCacheSize 700000
```

```
MCacheSize MCacheMaxObjectSize
```



| [FAQ](#) |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_mime

This translation may be out of date. Check the English version for recent changes.

```
┆  
┆ )  
┆ Base  
┆ mime_module  
┆ mod_mime.c
```

() (MIME

[AddCharset](#)

[AddEncoding](#)

[AddHandler](#)

[AddLanguage](#)

[AddType](#)

content-encoding, content-language, MIME (content-type)

[TypesConfig](#)

MIME

[mod\\_mime](#)

[AddHa](#)

[AddInputFilter](#)

[mod\\_negotiation](#) Multiviews

[mod\\_mime](#)

[core](#)

(,

[SetOutputFilter](#)

[mod\\_mime](#)

Last-Modified

'touch'

()

MimeMagicFile  
AddDefaultCharset  
ForceType  
DefaultType  
SetHandler  
SetInputFilter  
SetOutputFilter



---

```
welcome.html.fr          text/html
welcome.fr.html
image/gif .html MIME      text/html
welcome.gif.html  MIME      text/html
```

```
en, de Content-Type: text/html
```

```
MIME
.html MIME      text/html          wo
imap-file text/html MIME
mod imap
```



---

MIME  
UUencoding

[HTTP/1.1 RFC 14.11](#)

*Content-Encoding*  
*Content-Encoding*

( )

Microsoft Word

pkzip

.zip      pkzip

Apache

Content-encoding: pkzip



---

mime ( )  
MIME  
AddType ( MimeMagicFi  
AddInputFilter, AddOutputFilter

## Charset

Apache Content-Language

```
Content-Language: en, fr  
Content-Type: text/plain; charset=ISO-8859-1
```

charset



## AddCharset

```
:\
:\ AddCharset charset extension [extension] ...
:\ , , , .htaccess
:\ FileInfo
:\ Base
:\ mod_mime
```

AddCharset charset  
MIME charset

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

xxxx.ja.jis charset ISO-2022-JP  
(xxxx.jis.ja) AddCharset  
charset

*extension*

- [mod\\_negotiation](#)
- [AddDefaultCharset](#)



```

:~
:~ AddEncoding MIME-enc extension [extension]
...
:~ , , , .htaccess
:~ FileInfo
:~ Base
:~ mod_mime

```

AddEncoding

*extension*

```

AddEncoding x-gzip .gz
AddEncoding x-compress .Z

```

.gz x-gzip

x-zip x-compress  
x- Apache

compress deflate

*extension*



## AddHandler

```

:
: AddHandler handler-name extension [extension]
:   ...
: , , , .htaccess
: FileInfo
: Base
: mod_mime
```

*extension* [handler-name](#)  
".cgi" CGI

```
AddHandler cgi-script .cgi
```

httpd.conf ".cgi" CGI

*extension*

- [SetHandler](#)



```
:  
: AddInputFilter filter[:filter...] extension  
  [extension] ...  
: , , , .htaccess  
: FileInfo  
: Base  
: mod_mime  
: 2.0.26
```

AddInputFilter *extension* POST  
SetInputFilter

- RemoveInputFilter
- SetInputFilter



## AddLanguage

```
[:  
[: AddLanguage MIME-lang extension [extension]  
  ...  
[: , , , .htaccess  
[: FileInfo  
[: Base  
[: mod_mime
```

**AddLanguage** content langu  
*extension* MIME

```
AddEncoding x-compress .Z  
AddLanguage en .en  
AddLanguage fr .fr
```

xxxx.en.Z compress (  
language

```
AddLanguage en .en  
AddLanguage en-gb .en  
AddLanguage en-us .en
```

.en en-us

*extension*

- mod\_negotiation



## AddOutputFilter

```
:  
: AddOutputFilter filter [;filter...] extension  
  [extension] ...  
: , , , .htaccess  
: FileInfo  
: Base  
: mod_mime  
: 2.0.26
```

AddOutputFilter *extension*  
AddOutputFilterByType

.shtml SSI mod\_deflate

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

- RemoveOutputFilter
- SetOutputFilter



## AddType

```
:\  
:\ AddType MIME-type extension [extension] ...  
:\ , , , .htaccess  
:\ FileInfo  
:\ Base  
:\ mod_mime
```

### AddType

*extension*

MIME (

```
AddType image/gif .gif
```

MIME	<a href="#">TypesConfig</a>	Add
------	-----------------------------	-----

*extension*

- [DefaultType](#)
- [ForceType](#)





```
⌵ path_info mod_mime
⌵ ModMimeUsePathInfo On|Off
⌵ ModMimeUsePathInfo Off
⌵
⌵ Base
⌵ mod_mime
⌵ Apache 2.0.41
```

ModMimeUsePathInfo mod\_mime Off p  
URL

```
ModMimeUsePathInfo On
```

/bar (foo.shtml) ModMimeUsePathInfo Or  
/bar/foo.shtml mod\_mime  
AddOutputFilter INCLUDES ..shtml INC  
ModMimeUsePathInfo INCLUDES

- AcceptPathInfo



```

: MultiViews
: MultiviewsMatch
  Any|NegotiatedOnly|Filters|Handlers
  [Handlers|Filters]
: MultiviewsMatch NegotiatedOnly
: , , .htaccess
: FileInfo
: Base
: mod_mime
: 2.0.26

```

```

MultiviewsMatch          mod_negotiation  Multiviews :
Multiviews (             index.html)
index.html.fr  index.html.gz)

```

NegotiatedOnly

```

                    Multi
500          index.html.cgi 1000          index.html.pl
  .cgi              .asis              .d

```

```

  mod_mime          A
.bak

```

Multiviews

```

MultiviewsMatch Handlers Filters

```

- [Options](#)
- [mod\\_negotiation](#)



- [:](#)
- [: RemoveCharset \*extension\* \[\*extension\*\] ...](#)
- [: , , .htaccess](#)
- [: FileInfo](#)
- [: Base](#)
- [: mod\\_mime](#)
- [: 2.0.24](#)

### RemoveCharset

*extension*

```
RemoveCharset .html .shtml
```



```
RemoveEncoding extension [extension] ...  
,, .htaccess  
FileInfo  
Base  
mod_mime
```

### RemoveEncoding

```
/foo/.htaccess:  
AddEncoding x-gzip .gz  
AddType text/plain .asc  
<Files *.gz.asc>  
  RemoveEncoding .gz  
</Files>
```

foo.gz gzip

foo.gz.as

```
RemoveEncoding AddEncoding
```

*extension*



```
RemoveHandler extension [extension] ...  
,, .htaccess  
FileInfo  
Base  
mod_mime
```

## RemoveHandler

### **/foo/.htaccess:**

```
AddHandler server-parsed .html
```

### **/foo/bar/.htaccess:**

```
RemoveHandler .html
```

```
/foo/bar .html SSI( mod_include )
```

*extension*



```
RemoveInputFilter extension [extension] ...  
,, .htaccess  
FileInfo  
Base  
mod_mime  
2.0.26
```

RemoveInputFilter

*extension*

- [AddInputFilter](#)
- [SetInputFilter](#)



```
RemoveLanguage extension [extension] ...  
, , .htaccess  
FileInfo  
Base  
mod_mime  
2.0.24
```

RemoveLanguage

*extension*



## RemoveOutputFilter

```
RemoveOutputFilter extension [extension] ...  
,, .htaccess  
FileInfo  
Base  
mod_mime  
2.0.26
```

### RemoveOutputFilter

*extension*

```
RemoveOutputFilter shtml
```

- [AddOutputFilter](#)



## RemoveType

```
RemoveType extension [extension] ...  
,, .htaccess  
FileInfo  
Base  
mod_mime
```

RemoveType

MIME

```
/foo/.htaccess:  
RemoveType .cgi
```

/foo/ .cgi

Default

```
RemoveType  
AddType
```

*extension*



## TypesConfig

```
:_ mime.types
:_ TypesConfig file-path
:_ TypesConfig conf/mime.types
:_
:_ Base
:_ mod_mime
```

### TypesConfig MIME

IANA

<http://www.isi.edu/in-notes/iana/assignments/media-types/media-types>

mime.types

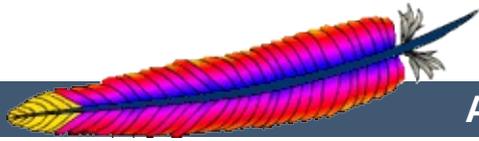
#### AddType

*MIME-type* [*extension*] ...

(*#*)

(1) IANA (2)  
Server Project category/x-subtype

- [mod\\_mime\\_magic](#)



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



## Apache HTTP Server Version 2.0

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

## Apache Module `mod_mime_magic`

<b>Description:</b>	Determines the MIME type of a file by looking at a few bytes of its contents
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>mime_magic_module</code>
<b>Source File:</b>	<code>mod_mime_magic.c</code>

### Summary

This module determines the MIME type of files in the same way the Unix `file(1)` command works: it looks at the first few bytes of the file. It is intended as a "second line of defense" for cases that `mod_mime` can't resolve.

This module is derived from a free version of the `file(1)` command for Unix, which uses "magic numbers" and other hints from a file's contents to figure out what the contents are. This module is active only if the magic file is specified by the `MimeMagicFile` directive.



The contents of the file are plain ASCII text in 4-5 columns. Blank lines are allowed but ignored. Commented lines use a hash mark (#). The remaining lines are parsed for the following columns:

Column	Description																						
1	byte number to begin checking from ">" indicates a dependency upon the previous non->" line																						
2	type of data to match <table border="1"> <tbody> <tr> <td>byte</td> <td>single character</td> </tr> <tr> <td>short</td> <td>machine-order 16-bit integer</td> </tr> <tr> <td>long</td> <td>machine-order 32-bit integer</td> </tr> <tr> <td>string</td> <td>arbitrary-length string</td> </tr> <tr> <td>date</td> <td>long integer date (seconds since Unix epoch/1970)</td> </tr> <tr> <td>beshort</td> <td>big-endian 16-bit integer</td> </tr> <tr> <td>belong</td> <td>big-endian 32-bit integer</td> </tr> <tr> <td>bedate</td> <td>big-endian 32-bit integer date</td> </tr> <tr> <td>leshort</td> <td>little-endian 16-bit integer</td> </tr> <tr> <td>lelong</td> <td>little-endian 32-bit integer</td> </tr> <tr> <td>ledate</td> <td>little-endian 32-bit integer date</td> </tr> </tbody> </table>	byte	single character	short	machine-order 16-bit integer	long	machine-order 32-bit integer	string	arbitrary-length string	date	long integer date (seconds since Unix epoch/1970)	beshort	big-endian 16-bit integer	belong	big-endian 32-bit integer	bedate	big-endian 32-bit integer date	leshort	little-endian 16-bit integer	lelong	little-endian 32-bit integer	ledate	little-endian 32-bit integer date
byte	single character																						
short	machine-order 16-bit integer																						
long	machine-order 32-bit integer																						
string	arbitrary-length string																						
date	long integer date (seconds since Unix epoch/1970)																						
beshort	big-endian 16-bit integer																						
belong	big-endian 32-bit integer																						
bedate	big-endian 32-bit integer date																						
leshort	little-endian 16-bit integer																						
lelong	little-endian 32-bit integer																						
ledate	little-endian 32-bit integer date																						
3	contents of data to match																						
4	MIME type if matched																						
5	MIME encoding if matched (optional)																						

For example, the following magic file lines would recognize some audio formats:

```
# Sun/NeXT audio data
```

```

0      string      .snd
>12   belong      1      audio/basic
>12   belong      2      audio/basic
>12   belong      3      audio/basic
>12   belong      4      audio/basic
>12   belong      5      audio/basic
>12   belong      6      audio/basic
>12   belong      7      audio/basic
>12   belong      23     audio/x-adpcm

```

Or these would recognize the difference between \* .doc files containing Microsoft Word or FrameMaker documents. (These are incompatible file formats which use the same file suffix.)

```

# Frame
0  string  \<MakerFile      application/x-frame
0  string  \<MIFFile      application/x-frame
0  string  \<MakerDictionary  application/x-frame
0  string  \<MakerScreenFon  application/x-frame
0  string  \<MML      application/x-frame
0  string  \<Book      application/x-frame
0  string  \<Maker      application/x-frame

# MS-Word
0  string  \376\067\0\043      application/msword
0  string  \320\317\021\340\241\261  application/msword
0  string  \333\245-\0\0\0      application/msword

```

An optional MIME encoding can be included as a fifth column. For example, this can recognize gzipped files and set the encoding for them.

```

# gzip (GNU zip, not to be confused with
#      [Info-ZIP/PKWARE] zip archiver)

0  string  \037\213  application/octet-stream  x-gzip

```



This module is not for every system. If your system is barely keeping up with its load or if you're performing a web server benchmark, you may not want to enable this because the processing is not free.

However, an effort was made to improve the performance of the original `file(1)` code to make it fit in a busy web server. It was designed for a server where there are thousands of users who publish their own documents. This is probably very common on intranets. Many times, it's helpful if the server can make more intelligent decisions about a file's contents than the file name allows ...even if just to reduce the "why doesn't my page work" calls when users improperly name their own files. You have to decide if the extra work suits your environment.



The following notes apply to the `mod_mime_magic` module and are included here for compliance with contributors' copyright restrictions that require their acknowledgment.

`mod_mime_magic`: MIME type lookup via file magic numbers  
Copyright (c) 1996-1997 Cisco Systems, Inc.

This software was submitted by Cisco Systems to the Apache Group in July 1997. Future revisions and derivatives of this source code must acknowledge Cisco Systems as the original contributor of this module. All other licensing and usage conditions are those of the Apache Group.

Some of this code is derived from the free version of the file command originally posted to comp.sources.unix. Copyright info for that program is included below as required.

- Copyright (c) Ian F. Darwin, 1987. Written by Ian F. Darwin.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must

not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

For compliance with Mr Darwin's terms: this has been very significantly modified from the free "file" command.

- all-in-one file for compilation convenience when moving from one version of Apache to the next.
- Memory allocation is done through the Apache API's pool structure.
- All functions have had necessary Apache API request or server structures passed to them where necessary to call other Apache API routines. (*i.e.*, usually for logging, files, or memory allocation in itself or a called function.)
- struct magic has been converted from an array to a single-ended linked list because it only grows one record at a time, it's only accessed sequentially, and the Apache API has no equivalent of `realloc()`.
- Functions have been changed to get their parameters from the server configuration instead of globals. (It should be reentrant now but has not been tested in a threaded environment.)
- Places where it used to print results to stdout now saves them in a list where they're used to set the MIME type in the Apache request record.
- Command-line flags have been removed since they will never be used here.

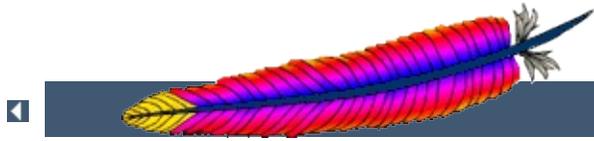


<b>Description:</b>	Enable MIME-type determination based on file contents using the specified magic file
<b>Syntax:</b>	MimeMagicFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_mime_magic

The `MimeMagicFile` directive can be used to enable this module, the default file is distributed at `conf/magic`. Non-rooted paths are relative to the `ServerRoot`. Virtual hosts will use the same file as the main server unless a more specific setting is used, in which case the more specific setting overrides the main server's file.

### Example

```
MimeMagicFile conf/magic
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_negotiation

This translation may be out of date. Check the English version for recent changes.

```
┆  
┆ Base  
┆ negotiation_module  
┆ mod_negotiation.c
```

- ( type-map ) variants
- MultiViews ( MultiViews [Option](#) )

[Options](#)

[mod\\_mime](#)



---

RFC 822

:

**Content-Encoding:**

Apache [AddEncoding](#) compress  
compress gzip x-gzip

**Content-Language:**

[\(RFC 1766\)](#)

**Content-Length:**

()

**Content-Type:**

MIME

**level**

text/html 2

**qs**

variant

0.0 1.0  
ASCII ASCII

```
Content-Type: image/jpeg; qs=0.8
```

**URI:**

() variant

uri.

**Body:**

Apache 2.0 Body

**Example:**

```
Body: ----xyz----  
<html>  
<body>  
<p>Content of the page.</p>
```

```
</body>  
</html>  
-----xyz-----
```



---

MultiViews

Multiviews Options

/some/dir/foo



## CacheNegotiatedDocs

```
CacheNegotiatedDocs On|Off  
CacheNegotiatedDocs Off  
,  
Base  
mod_negotiation  
2.0
```

HTTP/1.0

HTTP/1.1

2.0

CacheNegotiatedDocs

on



## ForceLanguagePriority

```
:  
: ForceLanguagePriority None|Prefer|Fallback  
  [Prefer|Fallback]  
: ForceLanguagePriority Prefer  
: , , .htaccess  
: FileInfo  
: Base  
: mod_negotiation  
: 2.0.30
```

### ForceLanguagePriority

```
ForceLanguagePriority Prefer HTTP 300  
(MULTIPLE CHOICES) LanguagePriority  
Accept-Language en de .500 ()
```

```
LanguagePriority en fr de  
ForceLanguagePriority Prefer
```

```
ForceLanguagePriority Fallback HTTP 406 (NOT  
ACCEPTABLE) LanguagePriority  
Language es variant  
variant
```

```
LanguagePriority en fr de  
ForceLanguagePriority Fallback
```

```
Prefer Fallback var  
variant vaiant
```

- [AddLanguage](#)



```
: variant
: LanguagePriority MIME-lang [MIME-lang] ...
: , , , .htaccess
: FileInfo
: Base
: mod_negotiation
```

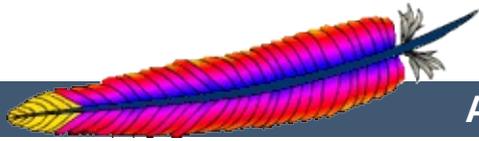
### LanguagePriority MultiViews

```
Example:
LanguagePriority en fr de
```

```
foo.html  foo.html.fr  foo.html.de
foo.html.fr
```

### [ForceLang](#)

- [AddLanguage](#)



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



## Apache HTTP Server Version 2.0

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

# Apache Module mod\_nw\_ssl

<b>Description:</b>	Enable SSL encryption for NetWare
<b>Status:</b>	Base
<b>Module Identifier:</b>	nwssl_module
<b>Source File:</b>	mod_nw_ssl.c
<b>Compatibility:</b>	NetWare only

## Summary

This module enables SSL encryption for a specified port. It takes advantage of the SSL encryption functionality that is built into the NetWare operating system.



<b>Description:</b>	List of additional client certificates
<b>Syntax:</b>	NWSSLTrustedCerts <i>filename</i> [ <i>filename</i> ] ...
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Specifies a list of client certificate files (DER format) that are used when creating a proxied SSL connection. Each client certificate used by a server must be listed separately in its own .der file.



<b>Description:</b>	Allows a connection to be upgraded to an SSL connection upon request
<b>Syntax:</b>	NWSSLUpgradeable [ <i>IP-address:</i> ]portnumber
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Allow a connection that was created on the specified address and/or port to be upgraded to an SSL connection upon request from the client. The address and/or port must have already be defined previously with a [Listen](#) directive.



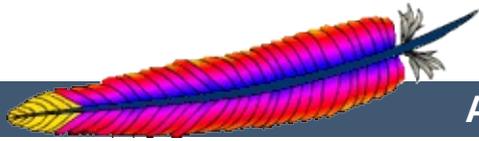
<b>Description:</b>	Enables SSL encryption for the specified port
<b>Syntax:</b>	<code>SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Specifies the port and the eDirectory based certificate name that will be used to enable SSL encryption. An optional third parameter also enables mutual authentication.

---

Copyright 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`

<b>Description:</b>	HTTP/1.1 proxy/gateway server
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_module</code>
<b>Source File:</b>	<code>mod_proxy.c</code>

## Summary

### Warning

Do not enable proxying with [ProxyRequests](#) until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

This module implements a proxy/gateway for Apache. It implements proxying capability for 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>
```



<b>Description:</b>	Ports that are allowed to CONNECT through the proxy
<b>Syntax:</b>	AllowCONNECT <i>port</i> [ <i>port</i> ] ...
<b>Default:</b>	AllowCONNECT 443 563
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `AllowCONNECT` directive specifies a list of port numbers to which the proxy CONNECT method may connect. Today's browsers use this method when a `https` connection is requested and proxy tunneling over HTTP is in effect.

By default, only the default `https` port (443) and the default `snews` port (563) are enabled. Use the `AllowCONNECT` directive to override this default and allow connections to the listed ports only.

Note that you'll need to have `mod_proxy_connect` present in the server in order to get the support for the CONNECT at all.



<b>Description:</b>	Hosts, domains, or networks that will be connected to directly
<b>Syntax:</b>	NoProxy <i>host</i> [ <i>host</i> ] ...
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is only useful for Apache proxy servers within intranets. The **NoProxy** directive specifies a list of subnets, IP addresses, hosts and/or domains, separated by spaces. A request to a host which matches one or more of these is always served directly, without forwarding to the configured **ProxyRemote** proxy server(s).

### Example

```
ProxyRemote * http://firewall.example.com:81
NoProxy .example.com 192.168.112.0/21
```

The *host* arguments to the **NoProxy** directive are one of the following type list:

### Domain

A *Domain* is a partially qualified DNS domain name, preceded by a period. It represents a list of hosts which logically belong to the same DNS domain or zone (*i.e.*, the suffixes of the hostnames are all ending in *Domain*).

### Examples

```
.com .apache.org.
```

To distinguish *Domains* from **Hostnames** (both syntactically and semantically; a DNS domain can have a DNS A record,

too!), *Domains* are always written with a leading period.

### Note

Domain name comparisons are done without regard to the case, and *Domains* are always assumed to be anchored in the root of the DNS tree, therefore 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](#)



## Proxy Directives

<b>Description:</b>	Container for directives applied to proxied resources
<b>Syntax:</b>	<code>&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

Directives placed in `<Proxy>` sections apply only to matching proxied content. Shell-style wildcards are allowed.

For example, the following will allow only hosts in `yournetwork.example.com` to access content via your proxy server:

```
<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from yournetwork.example.com
</Proxy>
```

The following example will process all files in the `foo` directory of `example.com` through the `INCLUDES` filter when they are sent through the proxy server:

```
<Proxy http://example.com/foo/*>
  SetOutputFilter INCLUDES
</Proxy>
```



## ProxyBadHeader Directive

<b>Description:</b>	Determines how to handle bad header lines in a response
<b>Syntax:</b>	ProxyBadHeader IsError   Ignore   StartBody
<b>Default:</b>	ProxyBadHeader IsError
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.44 and later

The **ProxyBadHeader** directive determines the 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

<b>Description:</b>	Default domain name for proxied requests
<b>Syntax:</b>	ProxyDomain <i>Domain</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is only useful for Apache proxy servers within intranets. The `ProxyDomain` directive specifies the default domain which the apache proxy server will belong to. If a request to a host without a domain name is encountered, a redirection response to the same host with the configured *Domain* appended will be generated.

### Example

```
ProxyRemote * http://firewall.example.com:81
NoProxy .example.com 192.168.112.0/21
ProxyDomain .example.com
```



<b>Description:</b>	Override error pages for proxied content
<b>Syntax:</b>	ProxyErrorOverride On Off
<b>Default:</b>	ProxyErrorOverride Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in version 2.0 and later

This directive is useful for reverse-proxy setups, where you want to have a common look and feel on the error pages seen by the end user. This also allows for included files (via mod\_include's SSI) to get the error code and act accordingly (default behavior would display the error page of the proxied server, turning this on shows the SSI Error message).



## ProxyFtpDirCharset Directive

<b>Description:</b>	Define the character set for proxied FTP listings
<b>Syntax:</b>	ProxyFtpDirCharset <i>character set</i>
<b>Default:</b>	ProxyFtpDirCharset ISO-8859-1
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.62 and later

The `ProxyFtpDirCharset` directive defines the character set to be set for FTP directory listings in HTML generated by [mod\\_proxy\\_ftp](#).



<b>Description:</b>	Determine size of internal data throughput buffer
<b>Syntax:</b>	ProxyIOBufferSize <i>bytes</i>
<b>Default:</b>	ProxyIOBufferSize 8192
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `ProxyIOBufferSize` directive adjusts the size of the internal buffer, which is used as a scratchpad for the data between input and output. The size must be less or equal 8192.

In almost every case there's no reason to change that value.



## ProxyMatch Directive

<b>Description:</b>	Container for directives applied to regular-expression-matched proxied resources
<b>Syntax:</b>	<code>&lt;ProxyMatch regex&gt; ...&lt;/ProxyMatch&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `<ProxyMatch>` directive is identical to the `<Proxy>` directive, except it matches URLs using regular expressions.



## ProxyMaxForwards Directive

<b>Description:</b>	Maximum number of proxies that a request can be forwarded through
<b>Syntax:</b>	ProxyMaxForwards <i>number</i>
<b>Default:</b>	ProxyMaxForwards 10
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0 and later

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

<b>Description:</b>	Maps remote servers into the local server URL-space
<b>Syntax:</b>	ProxyPass [ <i>path</i> ] !  <i>url</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. *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.



<b>Description:</b>	Adjusts the URL in HTTP response headers sent from a reverse proxied server
<b>Syntax:</b>	ProxyPassReverse [ <i>path</i> ] <i>url</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive lets Apache adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache is used as a reverse proxy 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

<b>Description:</b>	Use incoming Host HTTP request header for proxy request
<b>Syntax:</b>	ProxyPreserveHost On Off
<b>Default:</b>	ProxyPreserveHost Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.31 and later.

When enabled, this option will pass the Host: line from the incoming request to the proxied host, instead of the hostname specified in the proxypass 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

<b>Description:</b>	Remote proxy used to handle certain requests
<b>Syntax:</b>	ProxyRemote <i>match remote-server</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This defines remote proxies to this proxy. *match* is either the name of a URL-scheme that the remote server supports, or a partial URL for which the remote server should be used, or \* to indicate the server should be contacted for all requests. *remote-server* is a partial URL for the remote server. Syntax:

```
remote-server = scheme://hostname[:port]
```

*scheme* is effectively the protocol that should be used to communicate with the remote server; only http 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

<b>Description:</b>	Remote proxy used to handle requests matched by regular expressions
<b>Syntax:</b>	<code>ProxyRemoteMatch <i>regex remote-server</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `ProxyRemoteMatch` is identical to the `ProxyRemote` directive, except the first argument is a regular expression match against the requested URL.



## ProxyRequests Directive

<b>Description:</b>	Enables forward (standard) proxy requests
<b>Syntax:</b>	ProxyRequests On Off
<b>Default:</b>	ProxyRequests Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This allows or prevents Apache from functioning as a forward proxy server. (Setting ProxyRequests to Off does not disable use of the [ProxyPass](#) directive.)

In a typical reverse proxy 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.



<b>Description:</b>	Network timeout for proxied requests
<b>Syntax:</b>	ProxyTimeout <i>seconds</i>
<b>Default:</b>	ProxyTimeout 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.31 and later

This directive allows a user to specify a timeout on proxy requests. This is useful when you have a slow/buggy appserver which hangs, and you would rather just return a timeout and fail gracefully instead of waiting however long it takes the server to return.



<b>Description:</b>	Information provided in the <code>Via</code> HTTP response header for proxied requests
<b>Syntax:</b>	<code>ProxyVia On Off Full Block</code>
<b>Default:</b>	<code>ProxyVia Off</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_proxy</code>

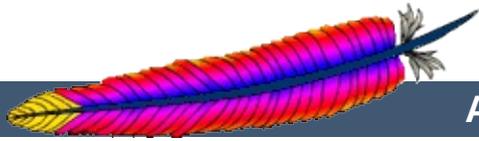
This directive controls the use of the `Via: HTTP` header by the proxy. Its intended use is to control the flow of proxy requests along a chain of proxy servers. See [RFC 2616](#) (HTTP/1.1), section 14.45 for an explanation of `Via: header` lines.

- If set to `Off`, which is the default, no special processing is performed. If a request or reply contains a `Via: header`, it is passed through unchanged.
- If set to `On`, each request and reply will get a `Via: header` line added for the current host.
- If set to `Full`, each generated `Via: header` line will additionally have the Apache server version shown as a `Via: comment` field.
- If set to `Block`, every proxy request will have all its `Via: header` lines removed. No new `Via: header` will be generated.

---

Copyright 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`

<b>Description:</b>	<code>mod_proxy</code> extension for CONNECT request handling
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_connect_module</code>
<b>Source File:</b>	<code>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

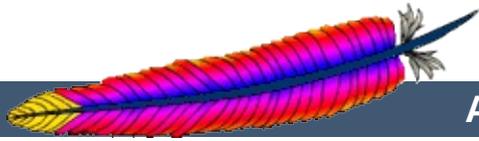
[AllowCONNECT](#)

[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`

<b>Description:</b>	FTP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_ftp_module</code>
<b>Source File:</b>	<code>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.

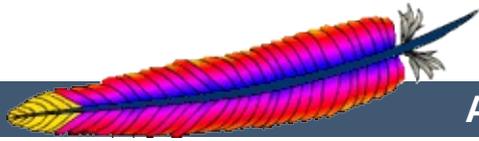
## See also

[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`

<b>Description:</b>	HTTP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_http_module</code>
<b>Source File:</b>	<code>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

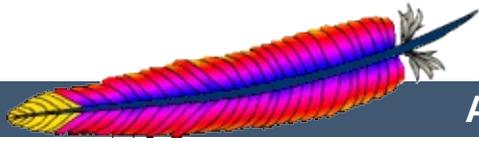
[mod\\_proxy](#)

[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`

<b>Description:</b>	Provides a rule-based rewriting engine to rewrite requested URLs on the fly
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>rewrite_module</code>
<b>Source File:</b>	<code>mod_rewrite.c</code>
<b>Compatibility:</b>	Available in Apache 1.3 and later

## Summary

This module uses a rule-based rewriting engine (based on a regular-expression parser) to rewrite requested URLs on the fly. It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule, to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests, of server variables, environment variables, HTTP headers, or time stamps. Even external database lookups in various formats can be used to achieve highly granular URL matching.

This module operates on the full URLs (including the path-info part) both in per-server context (`httpd.conf`) and per-directory context (`.htaccess`) and can generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to an internal proxy throughput.

Further details, discussion, and examples, are provided in the [detailed `mod\_rewrite` documentation](#).

## See also

[Rewrite Flags](#)



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).

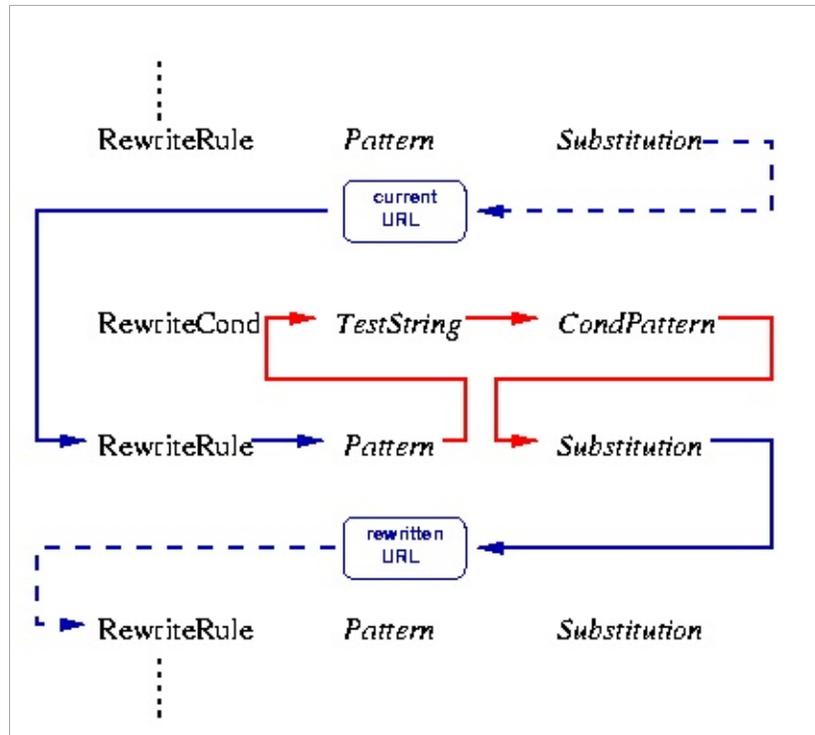
Once a request comes in, and Apache has determined the appropriate server (or virtual server), the rewrite engine starts the URL-to-filename translation, processing the `mod_rewrite` directives from the per-server configuration. A few steps later, when the final data directories are found, the per-directory 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.

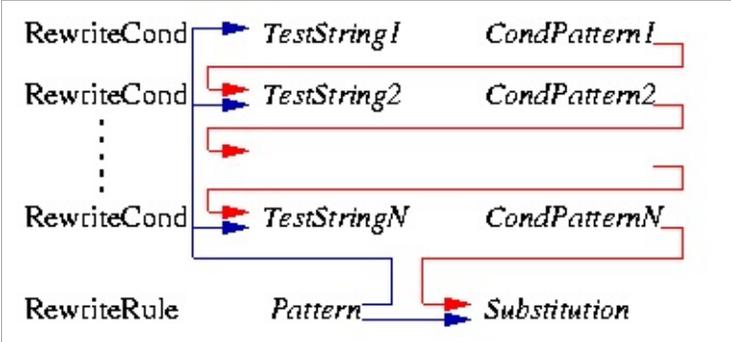
If **RewriteConds** exist, an inner loop is started, processing them in the order that they are listed. Conditions are not matched against the current URL directly. A *TestString* is constructed by expanding variables, back-references, map lookups, etc., against which the *CondPattern* is matched. If 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.



<b>Description:</b>	Sets the base URL for per-directory rewrites
<b>Syntax:</b>	RewriteBase <i>URL-path</i>
<b>Default:</b>	See usage for information.
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	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).



<b>Description:</b>	Defines a condition under which rewriting will take place
<b>Syntax:</b>	<code>RewriteCond <i>TestString CondPattern</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteCond` directive defines a rule condition. One or more `RewriteCond` can precede a `RewriteRule` directive. The following rule is then only used if both the current state of the URI matches its pattern, **and** if these conditions are met.

`TestString` is a string which can contain the following expanded constructs in addition to plain text:

- **RewriteRule backreferences:** These are backreferences of the form `$N` ( $0 \leq N \leq 9$ ), which provide access to the grouped parts (in parentheses) of the pattern, from the `RewriteRule` which is subject to the current set of `RewriteCond` conditions..
- **RewriteCond backreferences:** These are backreferences of the form `%N` ( $1 \leq N \leq 9$ ), which provide access to the grouped parts (again, in parentheses) of the pattern, from the last matched `RewriteCond` in the current set of conditions.
- **RewriteMap expansions:** These are expansions of the form `${mapname:key|default}`. See [the documentation for RewriteMap](#) for more details.
- **Server-Variables:** These are variables of the form `%{NAME_OF_VARIABLE}` where `NAME_OF_VARIABLE` can be a string taken from the following list:

---

**HTTP headers:**

**connection &**

	<b>request:</b>	
HTTP_USER_AGENT	REMOTE_ADDR	
HTTP_REFERER	REMOTE_HOST	
HTTP_COOKIE	REMOTE_PORT	
HTTP_FORWARDED	REMOTE_USER	
HTTP_HOST	REMOTE_IDENT	
HTTP_PROXY_CONNECTION	REQUEST_METHOD	
HTTP_ACCEPT	SCRIPT_FILENAME	
	PATH_INFO	
	QUERY_STRING	
	AUTH_TYPE	
<b>server internals:</b>	<b>system stuff:</b>	<b>special:</b>
DOCUMENT_ROOT	TIME_YEAR	API_VERSION
SERVER_ADMIN	TIME_MON	THE_REQUEST
SERVER_NAME	TIME_DAY	REQUEST_URI
SERVER_ADDR	TIME_HOUR	REQUEST_FILENAME
SERVER_PORT	TIME_MIN	IS_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.



## RewriteEngine Directive

<b>Description:</b>	Enables or disables runtime rewriting engine
<b>Syntax:</b>	RewriteEngine on off
<b>Default:</b>	RewriteEngine off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteEngine` directive enables or disables the runtime rewriting engine. If it is set to `off` this module does no runtime processing at all. It does not even update the `SCRIPT_URx` environment variables.

Use this directive to disable the module instead of commenting out all the `RewriteRule` directives!

Note that, 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`



<b>Description:</b>	Sets the name of the lock file used for <a href="#">RewriteMap</a> synchronization
<b>Syntax:</b>	RewriteLock <i>file-path</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

This directive sets the filename for a synchronization lockfile which mod\_rewrite needs to communicate with [RewriteMap programs](#). Set this lockfile to a local path (not on a NFS-mounted device) when you want to use a rewriting map-program. It is not required for other types of rewriting maps.



## RewriteLog Directive

<b>Description:</b>	Sets the name of the file used for logging rewrite engine processing
<b>Syntax:</b>	RewriteLog <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The **RewriteLog** directive sets the name of the file to which the server logs any rewriting actions it performs. If the name does not begin with a slash ('/') then it is assumed to be relative to the *Server Root*. The directive should occur only once per server config.

To disable the logging of rewriting actions it is not recommended to set *Filename* to `/dev/null`, because although the rewriting engine does not then output to a logfile it still creates the logfile output internally. **This will slow down the server with no advantage to the administrator!** To disable logging either remove or comment out the **RewriteLog** directive or use `RewriteLogLevel 0!`

### 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"
```



<b>Description:</b>	Sets the verbosity of the log file used by the rewrite engine
<b>Syntax:</b>	<code>RewriteLogLevel <i>Level</i></code>
<b>Default:</b>	<code>RewriteLogLevel 0</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_rewrite</code>

The `RewriteLogLevel` directive sets the verbosity level of the rewriting logfile. The default level 0 means no logging, while 9 or more means that practically all actions are logged.

To disable the logging of rewriting actions simply set *Level* to 0. This disables all rewrite action logs.

Using a high value for *Level* will slow down your Apache server dramatically! Use the rewriting logfile at a *Level* greater than 2 only for debugging!

### Example

```
RewriteLogLevel 3
```



<b>Description:</b>	Defines a mapping function for key-lookup
<b>Syntax:</b>	<code>RewriteMap MapName MapType:MapSource</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite
<b>Compatibility:</b>	The choice of different dbm types is available in Apache 2.0.41 and later

The `RewriteMap` directive defines a *Rewriting Map* which can be used inside rule substitution strings by the mapping-functions to insert/substitute fields through a key lookup. The source of this lookup can be of various types.

The *MapName* is the name of the map and will be used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

```
${ MapName : LookupKey }  
${ MapName : LookupKey | DefaultValue }
```

When such a construct occurs, the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the key is not found then it is substituted by *DefaultValue* or by the empty string if no *DefaultValue* was specified.

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!



<b>Description:</b>	Sets some special options for the rewrite engine
<b>Syntax:</b>	<code>RewriteOptions Options</code>
<b>Default:</b>	<code>RewriteOptions MaxRedirects=10</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite
<b>Compatibility:</b>	MaxRedirects is 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.



<b>Description:</b>	Defines rules for the rewriting engine
<b>Syntax:</b>	RewriteRule <i>Pattern Substitution</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite
<b>Compatibility:</b>	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:

<b>Text:</b>	
.	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
<b>Quantifiers:</b>	
?	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)
<b>Grouping:</b>	
(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 ( $\% \{ \text{VARNAME} \}$ )
4. [mapping-function](#) calls ( $\$ \{ \text{mapname} : \text{key} | \text{default} \}$ )

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`:**

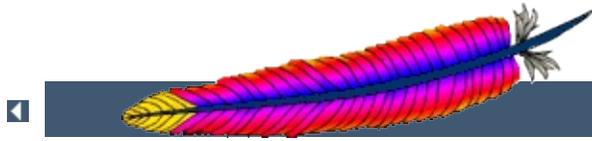
Given Rule	Resulting Substit
^/somepath(.*) otherpath\$1	invalid, not supp
^/somepath(.*) otherpath\$1 [R]	invalid, not supp
^/somepath(.*) otherpath\$1 [P]	invalid, not supp
^/somepath(.*) /otherpath\$1	/otherpath/pathir
^/somepath(.*) /otherpath\$1 [R]	http://thishost/c via external redi
^/somepath(.*) /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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_setenvif

This translation may be out of date. Check the English version for recent changes.

- [:](#)
- [:](#) Base
- [:](#) setenvif\_module
- [:](#) mod\_setenvif.c

## mod\_setenvif

```
BrowserMatch ^Mozilla netscape
BrowserMatch MSIE !netscape
```

[Apache](#)



```
⋮ HTTP User-Agent
⋮ BrowserMatch regex [!]env-variable[=value]
  [[!]env-variable[=value]] ...
⋮ , , , .htaccess
⋮ FileInfo
⋮ Base
⋮ mod_setenvif
```

BrowserMatch SetEnvIf

User-Agent HT

:

```
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
```



```
⌘ HTTP User-Agent
⌘ BrowserMatchNoCase regex [!]env-
  variable[=value] [[!]env-variable[=value]]
  ...
⌘ , , , .htaccess
⌘ FileInfo
⌘ Base
⌘ mod_setenvif
⌘ Apache 1.2 (Apache 1.2)
```

## BrowserMatchNoCase

## BrowserMatch

```
BrowserMatchNoCase mac platform=macintosh
BrowserMatchNoCase win platform=windows
```

## BrowserMatch

## BrowserMatchNoCase

## SetEnvIfNoCase

2 :

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```



```

:
: SetEnvIf attribute regex [!]env-
:   variable[=value] [[!]env-variable[=value]]
:   ...
: , , , .htaccess
: FileInfo
: Base
: mod_setenvif

```

## SetEnvIf

1. HTTP ( [RFC 2616](#) ) Host, User-Agent, Referer, Accept-Language
2. :
  - Remote\_Host - ()
  - Remote\_Addr - IP
  - Server\_Addr - IP (2.0.4)
  - Request\_Method - ( GET, POST )
  - Request\_Protocol -
  - Request\_URI - URL
3. `SetEnvIf` `SetEnvIf[NoCase]`  
`()`  
 ( *regex*) [Perl](#) POSIX.2 `egrep` *regex*  
*attribute*

1. *varname*

2. *!varname*

3. *varname=value*

"1"

*regex*

```
:  
SetEnvIf Request_URI "\.gif$" object_is_image=gif  
SetEnvIf Request_URI "\.jpg$" object_is_image=jpg  
SetEnvIf Request_URI "\.xbm$" object_is_image=xbm  
:  
SetEnvIf Referer www\.mydomain\.com intra_site_referral  
:  
SetEnvIf object_is_image xbm XBIT_PROCESSING=1  
:  
SetEnvIf ^TS* ^[a-z].* HAVE_TS
```

*object\_is\_image*

*intra\_site\_referral*

"TS" [a-z]

- [Apache](#)



```

:
: SetEnvIfNoCase attribute regex [!]env-
:   variable[=value] [[!]env-variable[=value]]
:   ...
: , , , .htaccess
: FileInfo
: Base
: mod_setenvif
: Apache 1.3

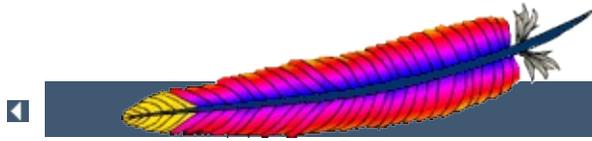
```

[SetEnvIfNoCase](#)

[SetEnvIf](#)

```
SetEnvIfNoCase Host Apache\.Org site=apache
```

```
HTTP Host: Apache.Org apache.org
site " apache"
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

## Apache mod\_so

- ┆
- ┆ Extension
- ┆ so\_module
- ┆ mod\_so.c
- ┆ Window ()
- Base

Unix

( .so )

Apache 1.3 Apache 2.0

— Apache



Apache 1.3.15 2.0 Windows  
mod\_so ApacheModuleFoo.dll

Apache API UNIX Windows V  
Windows Unix

Configure Unix Apache

DLL DLL  
Apache

DLL  
AP\_MODULE\_DECLARE\_DATA (Apache )

```
module foo_module;
```

```
module AP_MODULE_DECLARE_DATA foo_module;
```

Unix Windows  
DLL libhttpd.lib  
modules .dsp  
DLL modules



## LoadFile

```
LoadFile filename [filename] ...  
Extension  
mod_so
```

### LoadFile

:

```
LoadFile libexec/libxmlparse.so
```



## LoadModule

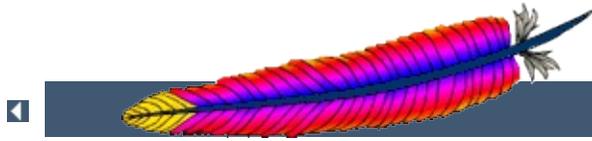
```
LoadModule module filename
           Extension
           mod_so
```

LoadModule *filename*

:

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

ServerRoot modules



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_speling

- URL
- Extension
- speling\_module
- mod\_speling.c

Apache

- 
- 
- 

Apache



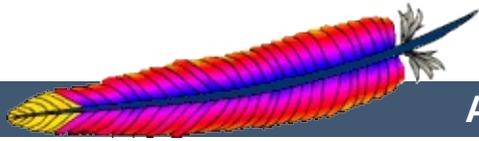
## CheckSpelling

```

: spelling
: CheckSpelling on|off
: CheckSpelling Off
: , , .htaccess
: Options
: Extension
: mod_speling
: CheckSpelling Apache 1.1 Apache 1.3
  Apache Apache 1.3.2

```

- 
- 
- (http://my.host/~apahce/ )
- <Location /st



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



## Apache HTTP Server Version 2.0

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

# Apache Module mod\_ssl

<b>Description:</b>	Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols
<b>Status:</b>	Extension
<b>Module Identifier:</b>	ssl_module
<b>Source File:</b>	mod_ssl.c

## Summary

This module provides SSL v2/v3 and TLS v1 support for the Apache HTTP Server. It was contributed by Ralf S. Engeschall based on his mod\_ssl project and originally derived from work by Ben Laurie.

This module relies on [OpenSSL](#) to provide the cryptography engine.

Further details, discussion, and examples are provided in the [SSL documentation](#).



This module 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.

<b>Variable Name:</b>	<b>Value Type:</b>	<b>Description:</b>
HTTPS	flag	HTTPS is being used.
SSL_PROTOCOL	string	The SSL protocol version (SSLv2, SSLv3, TLSv1)
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"
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	<code>SSLCACertificateFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLCACertificatePath](#).

### Example

```
SSLCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-client.crt
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	SSLCACertificatePath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose clients you deal with. These are used to verify the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value*.N. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCACertificatePath /usr/local/apache2/conf/ssl.crt/
```



**Description:** File of concatenated PEM-encoded CA CRLs for Client Auth

**Syntax:** SSLCARevocationFile *file-path*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLCARevocationPath](#).

### Example

```
SSLCARevocationFile /usr/local/apache2/conf/ssl.crl/ca-bundle-client.crl
```



<b>Description:</b>	Directory of PEM-encoded CA CRLs for Client Auth
<b>Syntax:</b>	SSLCARevocationPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose clients you deal with. These are used to revoke the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCARevocationPath /usr/local/apache2/conf/ssl.crl/
```



<b>Description:</b>	File of PEM-encoded Server CA Certificates
<b>Syntax:</b>	<code>SSLCertificateChainFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ssl</code>

This directive sets the optional *all-in-one* file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA Certificate files, usually in certificate chain order.

This should be used alternatively and/or additionally to [SSLCACertificatePath](#) for explicitly constructing the server certificate chain which is sent to the browser in addition to the server certificate. It is especially useful to avoid conflicts with CA certificates when using client authentication. Because although placing a CA certificate of the server certificate chain into [SSLCACertificatePath](#) has the same effect for the certificate chain construction, it has the side-effect that client certificates issued by this same CA certificate are also accepted on client authentication. 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
```



<b>Description:</b>	Server PEM-encoded X.509 Certificate file
<b>Syntax:</b>	SSLCertificateFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive points to the PEM-encoded 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
```



<b>Description:</b>	Server PEM-encoded Private Key file
<b>Syntax:</b>	SSLCertificateKeyFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive points to the PEM-encoded Private Key file for the server. If the 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
```



<b>Description:</b>	Cipher Suite available for negotiation in SSL handshake
<b>Syntax:</b>	SSLCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This complex directive uses a colon-separated *cipher-spec* string consisting of OpenSSL cipher specifications to configure the Cipher Suite the client is permitted to negotiate in the SSL handshake phase. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured Cipher Suite after the HTTP request was read but before the HTTP response is sent.

An SSL cipher specification in *cipher-spec* is composed of 4 major attributes plus a few extra minor ones:

- *Key Exchange Algorithm:*  
RSA or Diffie-Hellman variants.
- *Authentication Algorithm:*  
RSA, Diffie-Hellman, DSS or none.
- *Cipher/Encryption Algorithm:*  
DES, Triple-DES, RC4, RC2, IDEA or none.
- *MAC Digest Algorithm:*  
MD5, SHA or SHA1.

An SSL cipher can also be an export cipher and is either a SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3). To

specify which ciphers to use, one can either specify all the Ciphers, one at a time, or use aliases to specify the preference and order for the ciphers (see [Table 1](#)).

<b>Tag</b>	<b>Description</b>
<i>Key Exchange Algorithm:</i>	
kRSA	RSA key exchange
kDHR	Diffie-Hellman key exchange with RSA key
kDHd	Diffie-Hellman key exchange with DSA key
kEDH	Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)
<i>Authentication Algorithm:</i>	
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication
<i>Cipher Encoding Algorithm:</i>	
eNULL	No encoding
DES	DES encoding
3DES	Triple-DES encoding
RC4	RC4 encoding
RC2	RC2 encoding
IDEA	IDEA encoding
<i>MAC Digest Algorithm:</i>	
MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	SHA hash function
<i>Aliases:</i>	
SSLv2	all SSL version 2.0 ciphers
SSLv3	all SSL version 3.0 ciphers

TLSv1	all TLS version 1.0 ciphers
EXP	all export ciphers
EXPORT40	all 40-bit export ciphers only
EXPORT56	all 56-bit export ciphers only
LOW	all low strength ciphers (no export, single DES)
MEDIUM	all ciphers with 128 bit encryption
HIGH	all ciphers using Triple-DES
RSA	all ciphers using RSA key exchange
DH	all ciphers using Diffie-Hellman key exchange
EDH	all ciphers using Ephemeral Diffie-Hellman key exchange
ADH	all ciphers using Anonymous Diffie-Hellman key exchange
DSS	all ciphers using DSS authentication
NULL	all ciphers using no encryption

Now where this becomes interesting is that these can be put together to specify the order and ciphers you wish to use. To speed this up there are also aliases (SSLv2, SSLv3, TLSv1, EXP, LOW, MEDIUM, HIGH) for certain groups of ciphers. These tags can be joined together with prefixes to form the *cipher-spec*. Available prefixes are:

- none: add cipher to list
- +: move matching ciphers to the current location in list
- -: remove cipher from list (can be added later again)
- !: kill cipher from list completely (can **not** be added later again)

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



<b>Description:</b>	SSL Engine Operation Switch
<b>Syntax:</b>	SSLEngine on off
<b>Default:</b>	SSLEngine off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine. This is 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>
```



<b>Description:</b>	Option to prefer the server's cipher preference order
<b>Syntax:</b>	SSLHonorCipherOrder <i>flag</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in Apache 2.0.65 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
```



<b>Description:</b>	Option to enable support for insecure renegotiation
<b>Syntax:</b>	SSLInsecureRenegotiation <i>flag</i>
<b>Default:</b>	SSLInsecureRenegotiation off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.0.64 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.



<b>Description:</b>	Semaphore for internal mutual exclusion of operations
<b>Syntax:</b>	SSLMutex <i>type</i>
<b>Default:</b>	SSLMutex none
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures the SSL engine's semaphore (aka. lock) which is used for mutual exclusion of operations which have to be done in a synchronized way between the pre-forked Apache server processes. This directive can only be used in the global server context because it's only useful to have one global mutex. This directive is designed to closely match the [AcceptMutex](#) directive

The following Mutex *types* are available:

- none | no  
This is the default where no Mutex is used at all. Use it at your own risk. But because currently the Mutex is mainly used for synchronizing write access to the SSL Session Cache you can live without it as long as you accept a sometimes garbled Session Cache. So it's not recommended to leave this the default. Instead configure a real Mutex.
- posixsem  
This is an elegant Mutex variant where a Posix Semaphore is used when possible. It is only available when the underlying platform and APR supports it.
- sysvsem  
This is a somewhat elegant Mutex variant where a SystemV IPC Semaphore is used when possible. It is possible to "leak"

SystemV semaphores if processes crash before the semaphore is removed. It is only available when the underlying platform and APR supports it.

- `sem`

This directive tells the SSL Module to pick the "best" semaphore implementation available to it, choosing between Posix and SystemV IPC, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- `pthread`

This directive tells the SSL Module to use Posix thread mutexes. It is only available if the underlying platform and APR supports it.

- `fcntl:/path/to/mutex`

This is a portable Mutex variant where a physical (lock-)file and the `fcntl()` function are used as the Mutex. Always use a local disk filesystem for `/path/to/mutex` and never a file residing on a NFS- or AFS-filesystem. It is only available when the underlying platform and APR supports it. Note: Internally, the Process ID (PID) of the Apache parent process is automatically appended to `/path/to/mutex` to make it unique, so you don't have to worry about conflicts yourself. Notice that this type of mutex is not available under the Win32 environment. There you *have* to use the semaphore mutex.

- `flock:/path/to/mutex`

This is similar to the `fcntl:/path/to/mutex` method with the exception that the `flock()` function is used to provide file locking. It is only available when the underlying platform and APR supports it.

- `file:/path/to/mutex`

This directive tells the SSL Module to pick the "best" file locking implementation available to it, choosing between `fcntl` and `flock`, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- `default` | `yes`

This directive tells the SSL Module to pick the default locking implementation as determined by the platform and APR.

### Example

```
SSLMutex file:/usr/local/apache/logs/ssl_mutex
```



<b>Description:</b>	Configure various SSL engine run-time options
<b>Syntax:</b>	SSLOptions [+ -] <i>option</i> ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control various run-time options on a per-directory basis. Normally, if multiple `SSLOptions` could apply to a directory, then the most specific one is taken completely; the options are not merged. However if *all* the options on the `SSLOptions` directive are preceded by a plus (+) or minus (-) symbol, the options are merged. Any options preceded by a + are added to the options currently in force, and any options preceded by a - are removed from the options currently in force.

The available *options* are:

- `StdEnvVars`  
When this option is enabled, the standard set of SSL related CGI/SSI environment variables are created. This per default is disabled for performance reasons, because the information extraction step is a rather expensive operation. So one usually enables this option for CGI and SSI requests only.
- `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>
```



<b>Description:</b>	Type of pass phrase dialog for encrypted private keys
<b>Syntax:</b>	SSLPassPhraseDialog <i>type</i>
<b>Default:</b>	SSLPassPhraseDialog builtin
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

When Apache starts up it has to read the various Certificate (see [SSLCertificateFile](#)) and Private Key (see [SSLCertificateKeyFile](#)) files of the SSL-enabled virtual servers. Because for security reasons the Private Key files are usually encrypted, mod\_ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. This query can be done in two ways which can be configured by *type*:

- **builtin**  
This is the default where an interactive terminal dialog occurs at startup time just before Apache detaches from the terminal. Here the administrator has to manually enter the Pass Phrase for each encrypted Private Key file. Because a lot of SSL-enabled virtual hosts can be configured, the following reuse-scheme is used to minimize the dialog: When a Private Key file is encrypted, all known Pass Phrases (at the beginning there are none, of course) are tried. If one of those known Pass Phrases succeeds no dialog pops up for this particular Private Key file. If none succeeded, another Pass Phrase is queried on the terminal and remembered for the next round (where it perhaps can be reused).

This scheme allows mod\_ssl to be maximally flexible (because for N encrypted Private Key files you *can* use N different Pass Phrases - but then you have to enter all of

them, of course) while minimizing the terminal dialog (i.e. when you use a single Pass Phrase for all N Private Key files this Pass Phrase is queried only once).

- `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
```



<b>Description:</b>	Configure usable SSL protocol flavors
<b>Syntax:</b>	SSLProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProtocol all
<b>Context:</b>	server config, virtual host
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control the SSL protocol flavors mod\_ssl should use when establishing its server environment. Clients then can only connect with one of the provided protocols.

The available (case-insensitive) *protocols* are:

- SSLv2  
This is the Secure Sockets Layer (SSL) protocol, version 2.0. It is the original SSL protocol as designed by Netscape Corporation.
- 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
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Remote Server Auth
<b>Syntax:</b>	SSLProxyCACertificateFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCACertificatePath](#).

### Example

```
SSLProxyCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-remote-server.crt
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for Remote Server Auth
<b>Syntax:</b>	SSLProxyCACertificatePath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose remote servers you deal with. These are used to verify the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value*.N. And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

### Example

```
SSLProxyCACertificatePath /usr/local/apache2/conf/ssl.crt/
```



<b>Description:</b>	File of concatenated PEM-encoded CA CRLs for Remote Server Auth
<b>Syntax:</b>	SSLProxyCAREvocationFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCAREvocationPath](#).

### Example

```
SSLProxyCAREvocationFile /usr/local/apache2/conf/ssl.crl/ca-  
bundle-remote-server.crl
```



<b>Description:</b>	Directory of PEM-encoded CA CRLs for Remote Server Auth
<b>Syntax:</b>	SSLProxyCARevocationPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose remote servers you deal with. These are used to revoke the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with [mod\\_ssl](#) to accomplish this task.

### Example

```
SSLProxyCARevocationPath /usr/local/apache2/conf/ssl.crl/
```



<b>Description:</b>	Cipher Suite available for negotiation in SSL proxy h
<b>Syntax:</b>	SSLProxyCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

Equivalent to SSLCipherSuite, but for the proxy connection.  
Please refer to [SSLCipherSuite](#) for additional information.



## SSLProxyEngine Directive

<b>Description:</b>	SSL Proxy Engine Operation Switch
<b>Syntax:</b>	SSLProxyEngine on off
<b>Default:</b>	SSLProxyEngine off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine for proxy. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for proxy usage in a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for proxy image both for the main server and all configured virtual hosts.

### Example

```
<VirtualHost _default_:443>  
SSLProxyEngine on  
...  
</VirtualHost>
```



<b>Description:</b>	File of concatenated PEM-encoded client certificates and keys to be used by the proxy
<b>Syntax:</b>	SSLProxyMachineCertificateFile <i>filename</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the all-in-one file where you keep the certificates and keys used for authentication of the proxy server to remote servers.

This referenced file is simply the concatenation of the various PEM-encoded certificate files, in order of preference. Use this directive alternatively or additionally to SSLProxyMachineCertificatePath.

Currently there is no support for encrypted private keys

Example:

```
SSLProxyMachineCertificateFile  
/usr/local/apache2/conf/ssl.crt/proxy.pem
```



<b>Description:</b>	Directory of PEM-encoded client certificates and keys to be used by the proxy
<b>Syntax:</b>	SSLProxyMachineCertificatePath <i>directory</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the certificates and keys used for authentication of the proxy server to remote servers.

The files in this directory must be PEM-encoded and are accessed through hash filenames. Additionally, you must create symbolic links named *hash-value.N*. And you should always make sure this directory contains the appropriate symbolic links. 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/
```



<b>Description:</b>	Configure usable SSL protocol flavors for proxy usage
<b>Syntax:</b>	SSLProxyProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProxyProtocol all
<b>Context:</b>	server config, virtual host
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control the SSL protocol flavors mod\_ssl should use when establishing its server environment for proxy . It will only connect to servers using one of the provided protocols.

Please refer to [SSLProtocol](#) for additional information.



<b>Description:</b>	Type of remote server Certificate verification
<b>Syntax:</b>	SSLProxyVerify <i>level</i>
<b>Default:</b>	SSLProxyVerify none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	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

<b>Description:</b>	Maximum depth of CA Certificates in Remote Server Certificate verification
<b>Syntax:</b>	SSLProxyVerifyDepth <i>number</i>
<b>Default:</b>	SSLProxyVerifyDepth 1
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets how deeply mod\_ssl should verify before deciding that the 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
```



<b>Description:</b>	Pseudo Random Number Generator (PRNG) seeding source
<b>Syntax:</b>	SSLRandomSeed <i>context source [bytes]</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures one or more sources for seeding the Pseudo Random Number Generator (PRNG) in OpenSSL at startup time (*context* is `start up`) and/or just before a new SSL connection is established (*context* is `connect`). This directive can only be used in the global server context because the PRNG is a global facility.

The following *source* variants are available:

- `builtin`  
This is the always available builtin seeding source. 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	



<b>Description:</b>	Deny access when SSL is not used for the HTTP request
<b>Syntax:</b>	SSLRequireSSL
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive forbids access unless HTTP over SSL (i.e. HTTPS) is enabled for the current connection. This is very handy inside the SSL-enabled virtual host or directories for defending against configuration errors that expose stuff that should be protected. When this directive is present all requests are denied which are not using SSL.

### Example

```
SSLRequireSSL
```



<b>Description:</b>	Type of the global/inter-process SSL Session Cache
<b>Syntax:</b>	SSLSessionCache <i>type</i>
<b>Default:</b>	SSLSessionCache none
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures the storage type of the global/inter-process SSL Session Cache. This cache is an optional facility which speeds up parallel request processing. For requests to the same server process (via HTTP keep-alive), OpenSSL already caches the SSL session information locally. But because modern clients request inlined images and other data via parallel requests (usually up to four parallel requests are common) those requests are served by *different* pre-forked server processes. Here an inter-process cache helps to avoid unnecessary session handshakes.

The following 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)
```



<b>Description:</b>	Number of seconds before an SSL session expires in the Session Cache
<b>Syntax:</b>	SSLSessionCacheTimeout <i>seconds</i>
<b>Default:</b>	SSLSessionCacheTimeout 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

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
```



<b>Description:</b>	Variable name to determine user name
<b>Syntax:</b>	SSLUserName <i>varname</i>
<b>Context:</b>	server config, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in Apache 2.0.51 and later

This directive sets the "user" field in the Apache request object. This is used by lower modules to identify the user with a character string. In particular, this may cause the environment variable REMOTE\_USER to be set. The *varname* can be any of the [SSL environment variables](#).

### Example

```
SSLUserName SSL_CLIENT_S_DN_CN
```



<b>Description:</b>	Type of Client Certificate verification
<b>Syntax:</b>	SSLVerifyClient <i>level</i>
<b>Default:</b>	SSLVerifyClient none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the Certificate verification level for the Client Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification level after the HTTP request was read but before the HTTP response is sent.

The following levels are available for *level*:

- **none**: no client Certificate is required at all
- **optional**: the client *may* present a valid Certificate
- **require**: the client *has to* present a valid Certificate
- **optional\_no\_ca**: the client may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all browsers and level **optional\_no\_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

### Example

```
SSLVerifyClient require
```



<b>Description:</b>	Maximum depth of CA Certificates in Client Certificate verification
<b>Syntax:</b>	SSLVerifyDepth <i>number</i>
<b>Default:</b>	SSLVerifyDepth 1
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

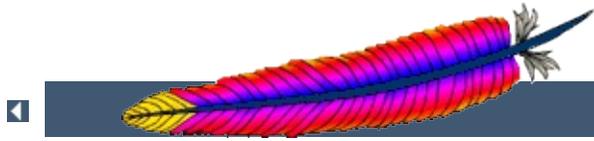
This directive sets how deeply mod\_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the client certificate. A depth of 0 means that self-signed client certificates are accepted only, the default depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under [SSLCACertificatePath](#)), etc.

### Example

```
SSLVerifyDepth 10
```

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_status

This translation may be out of date. Check the English version for recent changes.

[:](#)  
[:](#) Base  
[:](#) status\_module  
[:](#) mod\_status.c

## Status

:

- 
- (:) (\*)
- (\*)
- 
- 1 1 (\*)
- Apache CPU (\*)
- (\*)

"(\*)"



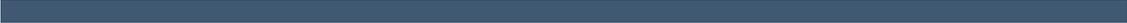
---

foo.com

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

<http://your.server.n>





N

status?refresh=N



---

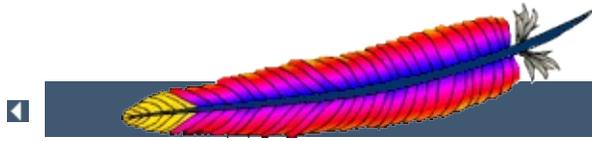
<http://your.server.name/server-status?auto>  
Apache /support

[mod\\_status](#)



## ExtendedStatus

```
:  
: ExtendedStatus On|Off  
: ExtendedStatus Off  
:  
: Base  
: mod_status  
: ExtendedStatus Apache 1.3.2
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_suexec

This translation may be out of date. Check the English version for recent changes.

- [: CGI](#)
- [: Extension](#)
- [: suexec\\_module](#)
- [: mod\\_suexec.c](#)
- [: Apache 2.0](#)

[suexec](#) CGI

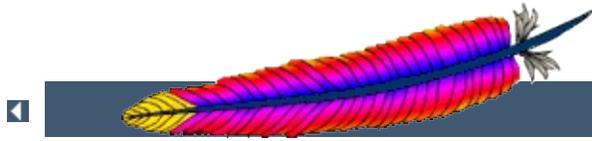
[SuEXEC](#)



```
: CGI  
: SuexecUserGroup User Group  
: ,  
: Extension  
: mod_suexec  
: SuexecUserGroup 2.0
```

SuexecUserGroup CGI CGI  
1.3 VirtualHosts User Group

```
SuexecUserGroup nobody nogroup
```



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

## Apache mod\_unique\_id

This translation may be out of date. Check the English version for recent changes.

[:](#)

[:](#) Extension

[:](#) unique\_id\_module

[:](#) mod\_unique\_id.c



---

## Apache Unix

(NTP )

- NTP
- IP

pid ( ID)

32

httpd

Unix (UTC 1970

1 1 ) 16

( *ip\_addr, pid, time\_stamp, counter* ) httpd 65536

pid

httpd

( ÷ 10) modulo 65536

0 )

pid pid

rand () seed seed

?

500

1.5%

UTC

NTP

UTC

UNIQUE\_ID 112 (32 IP

32 pid, 32 16

[A-Za-z0-9@-] MIME base64 19

base64

[A-Za-z0-9+ /]

+ / URL

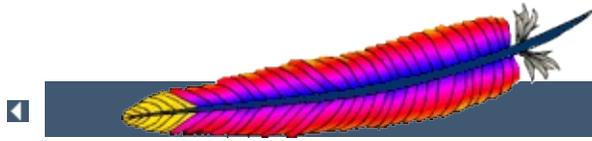
: IP pid,  
UNIQUE\_ID

UNIQUE\_ID

Window

)httpd  
)

(



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache 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

- 
- disabled            enabled ()
- disabled
- enabled

```

enabled    disabled            UserDir
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 ./" "~ro
disabled root"
```

:

UserDir :

```
UserDir disabled
UserDir enabled user1 user2 user3
```

UserDir :

```
UserDir enabled
UserDir disabled user4 user5 user6
```

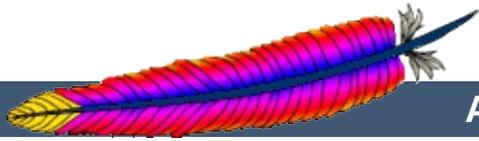
:

```
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
```

Apache

- [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

<b>Description:</b>	<i>Clickstream</i> logging of user activity on a site
<b>Status:</b>	Extension
<b>Module Identifier:</b>	usertrack_module
<b>Source File:</b>	mod_usertrack.c

### Summary

Previous releases of Apache have included a module which generates a 'clickstream' log of user activity on a site using cookies. This was called the "cookies" module, mod\_cookies. In Apache 1.2 and later this module has been renamed the "user tracking" module, mod\_usertrack. This module has been simplified and new directives added.



## Logging

Previously, the cookies module (now the user tracking module) did its own logging, using the `CookieLog` directive. In this release, this module does no logging at all. Instead, a configurable log format file should be used to log user click-streams. This is possible because the logging module now allows multiple log files. The cookie itself is logged by using the text `%{cookie}`n in the log file format. For example:

```
CustomLog logs/clickstream "%{cookie}n %r %t"
```

For backward compatibility the configurable log module implements the old `CookieLog` directive, but this should be upgraded to the above `CustomLog` directive.



(the following is from message  
<022701bda43d\$9d32bbb0\$1201a8c0@christian.office.sane.com>  
in the new-httpd archives)

From: "Christian Allen" <christian@sane.com>  
Subject: Re: Apache Y2K bug in mod\_usertrack.c  
Date: Tue, 30 Jun 1998 11:41:56 -0400

Did some work with cookies and dug up some info t

True, Netscape claims that the correct format NOW  
four digit dates do in fact work... for Netscape  
is. However, 3.x and below do NOT accept them.  
originally had a 2-digit standard, and then with  
probably a few complaints, changed to a four digit  
Fortunately, 4.x also understands the 2-digit form  
ensure that your expiration date is legible to th  
use 2-digit dates.

However, this does not limit expiration dates to  
an expiration year of "13", for example, it is in  
1913! In fact, you can use an expiration year of  
understood as "2037" by both MSIE and Netscape ve  
about versions previous to those). Not sure why  
particular year as its cut-off point, but my gues  
to UNIX's 2038 problem. Netscape/MSIE 4.x seem t  
2-digit years beyond that, at least until "50" fo  
understand up until about "70", but not for sure)

Summary: Mozilla 3.x and up understands two digi  
(2037). Mozilla 4.x understands up until at leas  
form, but also understands 4-digit years, which c  
9999. Your best bet for sending a long-life cook  
time late in the year "37".



<b>Description:</b>	The domain to which the tracking cookie applies
<b>Syntax:</b>	CookieDomain <i>domain</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive controls the setting of the domain to which the tracking cookie applies. If not present, no domain is included in the cookie header field.

The domain string **must** begin with a dot, and **must** include at least one embedded dot. That is, `.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`).



<b>Description:</b>	Expiry time for the tracking cookie
<b>Syntax:</b>	CookieExpires <i>expiry-period</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

When used, this directive sets an expiry time on the cookie generated by the usertrack module. The *expiry-period* can be given either as a number of seconds, or in the format such as "2 weeks 3 days 7 hours". Valid denominations are: years, months, weeks, days, hours, minutes and seconds. If the expiry time is in any format other than one number indicating the number of seconds, it must be enclosed by double quotes.

If this directive is not used, cookies last only for the current browser session.



<b>Description:</b>	Name of the tracking cookie
<b>Syntax:</b>	CookieName <i>token</i>
<b>Default:</b>	CookieName Apache
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive allows you to change the name of the cookie this module uses for its tracking purposes. By default the cookie is named "Apache".

You must specify a valid cookie name; results are unpredictable if you use a name containing unusual characters. Valid characters include A-Z, a-z, 0-9, "\_", and "-".



<b>Description:</b>	Format of the cookie header field
<b>Syntax:</b>	CookieStyle <i>Netscape   Cookie   Cookie2   RFC2109   RFC2961</i>
<b>Default:</b>	CookieStyle Netscape
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive controls the format of the cookie header field. The three formats allowed are:

- **Netscape**, which is the original but now deprecated syntax. This is the default, and the syntax Apache has historically used.
- **Cookie** or **RFC2109**, which is the syntax that superseded the Netscape syntax.
- **Cookie2** or **RFC2965**, which is the most current cookie syntax.

Not all clients can understand all of these formats. but you should use the newest one that is generally acceptable to your users' browsers. At the time of writing, most browsers only fully support CookieStyle Netscape.



## CookieTracking Directive

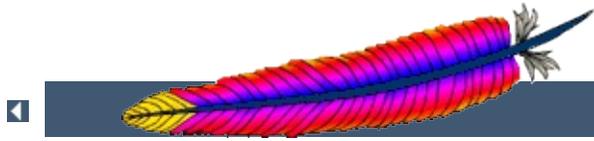
<b>Description:</b>	Enables tracking cookie
<b>Syntax:</b>	CookieTracking on off
<b>Default:</b>	CookieTracking off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

When [mod\\_usertrack](#) is loaded, and `CookieTracking on` is set, Apache will send a user-tracking cookie for all new requests. This directive can be used to turn this behavior on or off on a per-server or per-directory basis. By default, enabling [mod\\_usertrack](#) will **not** activate cookies.

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

# Apache mod\_version

This translation may be out of date. Check the English version for recent changes.

- [:](#)
- [:](#) Extension
- [:](#) version\_module
- [:](#) mod\_version.c
- [:](#) 2.0.54

httpd

```
<IfVersion 2.1.0>  
  # current httpd version is exactly 2.1.0  
</IfVersion>  
  
<IfVersion >= 2.2>  
  # use really new features :-)  
</IfVersion>
```



OPERATOR

```

:
: <IfVersion [[!]operator] version> ...
:   </IfVersion>
: , , , .htaccess
: All
: Extension
: mod_version

```

```

<IfVersion> httpd
major[.minor[.patch]]          2.1.0  2.2
patch                          0

```

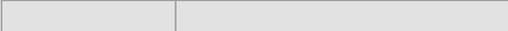
<i>operator</i>	
= ==	httpd
>	httpd
>=	httpd
<	httpd
<=	httpd

```

<IfVersion >= 2.1>
  # this happens only in versions greater or
  # equal 2.1.0.
</IfVersion>

```

http :



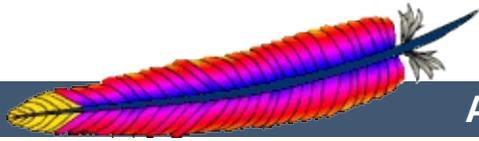
<i>operator</i>	
= or ==	<i>version</i> <i>/regex/</i>
~	<i>version</i> <i>regex</i>

```
<IfVersion = /^2.1.[01234]$/>
  # e.g. workaround for buggy versions
</IfVersion>
```

( !):

```
<IfVersion !~ ^2.1.[01234]$/>
  # not for those versions
</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`

<b>Description:</b>	Provides for dynamically configured mass virtual hosting
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>vhost_alias_module</code>
<b>Source File:</b>	<code>mod_vhost_alias.c</code>

## Summary

This module creates dynamically configured virtual hosts, by allowing the IP address and/or the `Host :` header of the HTTP request to be used as part of the pathname to determine what files to serve. This allows for easy use of a huge number of virtual hosts with similar configurations.

### Note

If `mod_alias` or `mod_userdir` are used for translating URIs to filenames, they will override the directives of `mod_vhost_alias` described below. For example, the following configuration will map `/cgi-bin/script.pl` to `/usr/local/apache2/cgi-bin/bin/script.pl` in all cases:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/  
VirtualScriptAlias /never/found/%0/cgi-bin/
```

## See also

[UseCanonicalName](#)

[Dynamically configured mass virtual hosting](#)



## Directory Name Interpolation

All the directives in this module interpolate a string into a pathname. The interpolated string (henceforth called the "name") may be either the server name (see the [UseCanonicalName](#) directive for details on how this is determined) or the IP address of the virtual host on the server in dotted-quad format. The interpolation is controlled by specifiers inspired by `printf` which have a number of formats:

<code>%%</code>	insert a %
<code>%p</code>	insert the port number of the virtual host
<code>%N.M</code>	insert (part of) the name

N and M are used to specify substrings of the name. N selects from the dot-separated components of the name, and M selects characters within whatever N has selected. M is optional and defaults to zero if it isn't present; the dot must be present if and only if M is present. The interpretation is as follows:

<code>0</code>	the whole name
<code>1</code>	the first part
<code>2</code>	the second part
<code>-1</code>	the last part
<code>-2</code>	the penultimate part
<code>2+</code>	the second and all subsequent parts
<code>-2+</code>	the penultimate and all preceding parts
<code>1+</code> and <code>-1+</code>	the same as <code>0</code>

If N or M is greater than the number of parts available a single underscore is interpolated.



For simple name-based virtual hosts you might use the following directives in your server configuration file:

```
UseCanonicalName Off
VirtualDocumentRoot /usr/local/apache/vhosts/%0
```

A request for  
`http://www.example.com/directory/file.html` will be satisfied by the file  
`/usr/local/apache/vhosts/www.example.com/directory`

For a very large number of virtual hosts it is a good idea to arrange the files to reduce the size of the vhosts directory. To do this you might use the following in your configuration file:

```
UseCanonicalName Off
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

A request for  
`http://www.domain.example.com/directory/file.html`  
will be satisfied by the file  
`/usr/local/apache/vhosts/example.com/d/o/m/domain`

A more even spread of files can be achieved by hashing from the end of the name, for example:

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/n/i/a/domain`

Alternatively you might use:

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/d/o/m/ain/di`

For IP-based virtual hosting you might use the following in your configuration file:

```
UseCanonicalName DNS
VirtualDocumentRootIP /usr/local/apache/vhosts/%1/%2/%3/%4/docs
VirtualScriptAliasIP /usr/local/apache/vhosts/%1/%2/%3/%4/cgi-
bin
```

A request for  
`http://www.domain.example.com/directory/file.html`  
would be satisfied by the file  
`/usr/local/apache/vhosts/10/20/30/40/docs/directo`  
if the IP address of `www.domain.example.com` were  
10.20.30.40. A request for  
`http://www.domain.example.com/cgi-bin/script.pl`  
would be satisfied by executing the program  
`/usr/local/apache/vhosts/10/20/30/40/cgi-  
bin/script.pl`.

If you want to include the `.` character in a  
`VirtualDocumentRoot` directive, but it clashes with a `%`  
directive, you can work around the problem in the following way:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%2.0.%3.0
```

A request for  
`http://www.domain.example.com/directory/file.html`  
will be satisfied by the file  
`/usr/local/apache/vhosts/domain.example/directory`

The LogFormat directives %V and %A are useful in conjunction with this module.



<b>Description:</b>	Dynamically configure the location of the document root for a given virtual host
<b>Syntax:</b>	<code>VirtualDocumentRoot <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualDocumentRoot none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualDocumentRoot` directive allows you to determine where Apache will find your documents based on the value of the server name. The result of expanding *interpolated-directory* is used as the root of the document tree in a similar manner to the `DocumentRoot` directive's argument. If *interpolated-directory* is none then `VirtualDocumentRoot` is turned off. This directive cannot be used in the same context as `VirtualDocumentRootIP`.



<b>Description:</b>	Dynamically configure the location of the document root for a given virtual host
<b>Syntax:</b>	<code>VirtualDocumentRootIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualDocumentRootIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualDocumentRootIP` directive is like the `VirtualDocumentRoot` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.



## VIRTUALSCRIPTALIAS DIRECTIVE

<b>Description:</b>	Dynamically configure the location of the CGI directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAlias <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAlias none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualScriptAlias` directive allows you to determine where Apache will find CGI scripts in a similar manner to `VirtualDocumentRoot` does for other documents. It matches requests for URIs starting `/cgi-bin/`, much like `ScriptAlias /cgi-bin/` would.



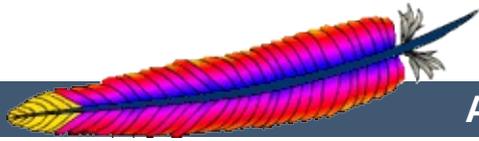
<b>Description:</b>	Dynamically configure the location of the cgi directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAliasIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAliasIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualScriptAliasIP` directive is like the `VirtualScriptAlias` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.

---

Copyright 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 `srml.conf`, which actually behaves for most purposes almost exactly like `<Directory />`).

Finally, after having served a request which involved reading `.htaccess` files, we need to discard the storage allocated for handling them. That is solved the same way it is solved wherever else similar problems come up, by tying those structures to the per-transaction resource pool.

## Per-directory configuration structures

Let's look out how all of this plays out in `mod_mime.c`, which defines the file typing handler which emulates the NCSA server's behavior of determining file types from suffixes. What we'll be looking at, here, is the code which implements the `AddType` and `AddEncoding` commands. These commands can appear in `.htaccess` files, so they must be handled in the module's private per-directory data, which in fact, consists of two separate tables for MIME types and encoding information, and is declared as follows:

```
typedef struct {
    table *forced_types;      /* Additional AddTyped stuff */
    table *encoding_types;   /* Added with AddEncoding... */
} mime_dir_config;
```

When the server is reading a configuration file, or `<Directory>` section, which includes one of the MIME module's commands, it needs to create a `mime_dir_config` structure, so those commands have something to act on. It does this by invoking the function it finds in the module's 'create per-dir config slot', with two arguments: the name of the directory to which this configuration

information applies (or NULL for `srm.conf`), and a pointer to a resource pool in which the allocation should happen.

(If we are reading a `.htaccess` file, that resource pool is the per-request resource pool for the request; otherwise it is a resource pool which is used for configuration data, and cleared on restarts. Either way, it is important for the structure being created to vanish when the pool is cleared, by registering a cleanup on the pool if necessary).

For the MIME module, the per-dir config creation function just `ap_pallocs` the structure above, and creates a couple of tables to fill it. That looks like this:

```
void *create_mime_dir_config (pool *p, char *dummy)
{
    mime_dir_config *new =
        (mime_dir_config *) ap_palloc (p,
        sizeof(mime_dir_config));

    new->forced_types = ap_make_table (p, 4);
    new->encoding_types = ap_make_table (p, 4);

    return new;
}
```

Now, suppose we've just read in a `.htaccess` file. We already have the per-directory configuration structure for the next directory up in the hierarchy. If the `.htaccess` file we just read in didn't have any [AddType](#) or [AddEncoding](#) commands, its per-directory config structure for the MIME module is still valid, and we can just use it. Otherwise, we need to merge the two structures somehow.

To do that, the server invokes the module's per-directory config merge function, if one is present. That function takes three arguments: the two structures being merged, and a resource pool in which to allocate the result. For the MIME module, all that needs

to be done is overlay the tables from the new per-directory config structure with those from the parent:

```
void *merge_mime_dir_configs (pool *p, void *parent_dirv, void
*subdirv)
{
    mime_dir_config *parent_dir = (mime_dir_config
*)parent_dirv;
    mime_dir_config *subdir = (mime_dir_config *)subdirv;
    mime_dir_config *new =
        (mime_dir_config *)ap_palloc (p, sizeof(mime_dir_config));

    new->forced_types = ap_overlay_tables (p, subdir-
>forced_types,
        parent_dir->forced_types);
    new->encoding_types = ap_overlay_tables (p, subdir-
>encoding_types,
        parent_dir->encoding_types);

    return new;
}
```

As a note -- if there is no per-directory merge function present, the server will just use the subdirectory's configuration info, and ignore the parent's. For some modules, that works just fine (e.g., for the includes module, whose per-directory configuration information consists solely of the state of the XBITHACK), and for those modules, you can just not declare one, and leave the corresponding structure slot in the module itself NULL.

## Command handling

Now that we have these structures, we need to be able to figure out how to fill them. That involves processing the actual [AddType](#) and [AddEncoding](#) commands. To find commands, the server looks in the module's command table. That table contains information on how many arguments the commands take, and in what formats, where it is permitted, and so forth. That information is sufficient to allow the server to invoke most command-handling functions with pre-parsed arguments. Without further ado, let's

look at the [AddType](#) command handler, which looks like this (the [AddEncoding](#) command looks basically the same, and won't be shown here):

```
char *add_type(cmd_parms *cmd, mime_dir_config *m, char *ct,
char *ext)
{
    if (*ext == '.') ++ext;
    ap_table_set (m->forced_types, ext, ct);
    return NULL;
}
```

This command handler is unusually simple. As you can see, it takes four arguments, two of which are pre-parsed arguments, the third being the per-directory configuration structure for the module in question, and the fourth being a pointer to a `cmd_parms` structure. That structure contains a bunch of arguments which are frequently of use to some, but not all, commands, including a resource pool (from which memory can be allocated, and to which cleanups should be tied), and the (virtual) server being configured, from which the module's per-server configuration data can be obtained if required.

Another way in which this particular command handler is unusually simple is that there are no error conditions which it can encounter. If there were, it could return an error message instead of `NULL`; this causes an error to be printed out on the server's `stderr`, followed by a quick exit, if it is in the main config files; for a `.htaccess` file, the syntax error is logged in the server error log (along with an indication of where it came from), and the request is bounced with a server error response (HTTP error status, code 500).

The MIME module's command table has entries for these commands, which look like this:

```

command_rec mime_cmds[] = {
    { "AddType", add_type, NULL, OR_FILEINFO, TAKE2,
      "a mime type followed by a file extension" },
    { "AddEncoding", add_encoding, NULL, OR_FILEINFO, TAKE2,
      "an encoding (e.g., gzip), followed by a file extension"
    },
    { NULL }
};

```

The entries in these tables are:

- The name of the command
- The function which handles it
- a (`void *`) pointer, which is passed in the `cmd_parms` structure to the command handler --- this is useful in case many similar commands are handled by the same function.
- A bit mask indicating where the command may appear. There are mask bits corresponding to each `AllowOverride` option, and an additional mask bit, `RSRC_CONF`, indicating that the command may appear in the server's own config files, but *not* in any `.htaccess` file.
- A flag indicating how many arguments the command handler wants pre-parsed, and how they should be passed in. `TAKE2` indicates two pre-parsed arguments. Other options are `TAKE1`, which indicates one pre-parsed argument, `FLAG`, which indicates that the argument should be `On` or `Off`, and is passed in as a boolean flag, `RAW_ARGS`, which causes the server to give the command the raw, unparsed arguments (everything but the command name itself). There is also `ITERATE`, which means that the handler looks the same as `TAKE1`, but that if multiple arguments are present, it should be called multiple times, and finally `ITERATE2`, which indicates that the command handler looks like a `TAKE2`, but if more arguments are present, then it should be called multiple times, holding the first argument constant.

- Finally, we have a string which describes the arguments that should be present. If the arguments in the actual config file are not as required, this string will be used to help give a more specific error message. (You can safely leave this NULL).

Finally, having set this all up, we have to use it. This is ultimately done in the module's handlers, specifically for its file-typing handler, which looks more or less like this; note that the per-directory configuration structure is extracted from the request\_rec's per-directory configuration vector by using the ap\_get\_module\_config function.

```
int find_ct(request_rec *r)
{
    int i;
    char *fn = ap_pstrdup (r->pool, r->filename);
    mime_dir_config *conf = (mime_dir_config *)
        ap_get_module_config(r->per_dir_config, &mime_module);
    char *type;

    if (S_ISDIR(r->finfo.st_mode)) {
        r->content_type = DIR_MAGIC_TYPE;
        return OK;
    }

    if((i=ap_rind(fn, '.')) < 0) return DECLINED;
    ++i;

    if ((type = ap_table_get (conf->encoding_types, &fn[i])))
    {
        r->content_encoding = type;

        /* go back to previous extension to try to use it as a
        type */
        fn[i-1] = '\0';
        if((i=ap_rind(fn, '.')) < 0) return OK;
        ++i;
    }

    if ((type = ap_table_get (conf->forced_types, &fn[i])))
    {
        r->content_type = type;
    }
}
```

```
    return OK;
}
```

## Side notes -- per-server configuration, virtual servers, etc.

The basic ideas behind per-server module configuration are basically the same as those for per-directory configuration; there is a creation function and a merge function, the latter being invoked where a virtual server has partially overridden the base server configuration, and a combined structure must be computed. (As with per-directory configuration, the default if no merge function is specified, and a module is configured in some virtual server, is that the base configuration is simply ignored).

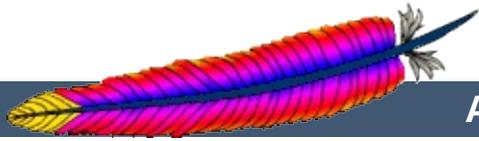
The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the `alias` module, which also indicates how a syntax error can be returned (note that the per-directory configuration argument to the command handler is declared as a dummy, since the module doesn't actually have per-directory config data):

```
char *add_redirect(cmd_parms *cmd, void *dummy, char *f, char
*url)
{
    server_rec *s = cmd->server;
    alias_server_conf *conf = (alias_server_conf *)
        ap_get_module_config(s->module_config, &alias_module);
    alias_entry *new = ap_push_array (conf->redirects);

    if (!ap_is_url (url)) return "Redirect to non-URL";

    new->fake = f; new->real = url;
    return NULL;
}
```

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



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



## Apache HTTP Server Version 2.0

[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 `0xa5s` 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.

	<b>ALLOC DEBUG</b>	<b>ALLOC USE MALLOC</b>	<b>POOL DEBUG</b>	<b>MAKE TABLE PROFILE</b>	<b>ALLOC STATS</b>
<b>ALLOC DEBUG</b>	-	No	Yes	Yes	Yes
<b>ALLOC USE MALLOC</b>	No	-	No	No	No
<b>POOL DEBUG</b>	Yes	No	-	Yes	Yes
<b>MAKE TABLE PROFILE</b>	Yes	No	Yes	-	Yes
<b>ALLOC STATS</b>	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.

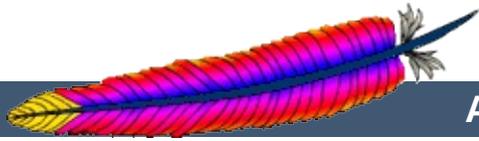
### Note

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 `&gt;` rather than `>`):

```
/**
 * return the final element of the pathname
 * @param pathname The path to get the final element of
 * @return the final element of the path
 * @tip Examples:
 * <pre>
 * "/foo/bar/gum" -&gt; "gum"
 * "/foo/bar/gum/" -&gt; ""
 * "gum" -&gt; "gum"
 * "wi\\n32\\stuff" -&gt; "stuff"
 * </pre>
 * @deffunc const char * ap_filename_of_pathname(const char
 *pathname)
 */
```

At the top of the header file, always include:

```
/**
 * @package Name of library header
 */
```

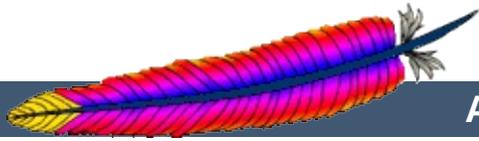
Doxygen uses a new HTML file for each package. The HTML files are named `{Name_of_library_header}.html`, so try to be concise with your names.

For a further discussion of the possibilities please refer to [the Doxygen site](#).

---

Copyright 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:

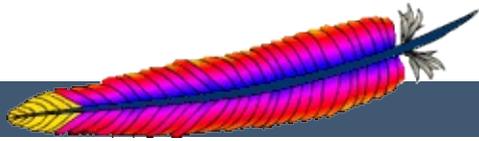
```
static void register_hooks()
{
    static const char * const aszPre[] = { "mod_xyz.c",
        "mod_abc.c", NULL };

    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](#)



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) > [Developer Documentation](#)

## Apache 1.3 Apache 2.0

`mod_mmap_static` Apache 2.0



---

apr\_status\_t  
ARP\_SUCCESS

apr\_status\_t

- apr\_pool\_t \*p
- apr\_pool\_t \*plog
- apr\_pool\_t \*ptemp
- server\_rec \*s

### APR

- pool becomes apr\_pool\_t
- table becomes apr\_table\_t



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);
};
```

post\_config (

?

```
ap_hook_phase_name(function_name, predecessors, successors,
position);
```

...

- HOOK\_FIRST
- HOOK\_MIDDLE
- HOOK\_LAST

mod\_mmap\_static    post\_config  
mmap\_static\_xlat core

aszPre

```

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 */
};

```

...

```

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 */
};

```

:

```

/* */
/* */
/* */

```

```
    /* */  
/* */  
    /* */  
/* */  
/* */  
    /* apr_table_t */  
/* */  
    /* */
```

...

```
ap_hook_post_config  
    ( _init )  
ap_hook_http_method  
    ( HTTP () )  
ap_hook_open_logs  
    ()  
ap_hook_auth_checker  
    ()  
ap_hook_access_checker  
    ()  
ap_hook_check_user_id  
    ( ID )  
ap_hook_default_port  
    ()  
ap_hook_pre_connection  
    ( accept )  
ap_hook_process_connection  
    ()
```

**ap\_hook\_child\_init**  
()

**ap\_hook\_create\_request**  
(??)

**ap\_hook\_fixups**  
()

**ap\_hook\_handler**  
()

**ap\_hook\_header\_parser**  
(post\_read\_request)

**ap\_hook\_insert\_filter**  
()

**ap\_hook\_log\_transaction**  
()

**ap\_hook\_optional\_fn\_retrieve**  
()

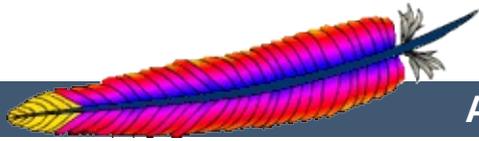
**ap\_hook\_post\_read\_request**  
()

**ap\_hook\_quick\_handler**

**ap\_hook\_translate\_name**  
(URI)

**ap\_hook\_type\_checker**  
()



[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

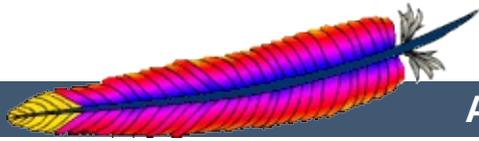
This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

### **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.

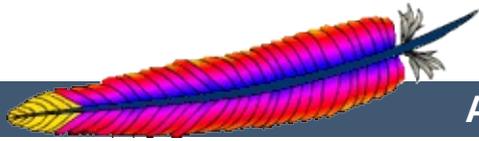
This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.



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.



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



## Apache HTTP Server Version 2.0

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

## Glossary

This glossary defines some of the common terminology related to Apache in particular, and web serving in general. More information on each concept is provided in the links.



## Access Control

The restriction of access to network realms. In an Apache context usually the restriction of access to certain *URLs*.

See: [Authentication, Authorization, and Access Control](#)

## Algorithm

An unambiguous formula or set of rules for solving a problem in a finite number of steps. Algorithms for encryption are usually called *Ciphers*.

## APache eXtension Tool (apxs)

A perl script that aids in compiling → [module](#) sources into Dynamic Shared Objects (→ [DSOs](#)) and helps install them in the Apache Web server.

See: Manual Page: [apxs](#)

## Authentication

The positive identification of a network entity such as a server, a client, or a user.

See: [Authentication, Authorization, and Access Control](#)

## Certificate

A data record used for authenticating network entities such as a server or a client. A certificate contains X.509 information pieces about its owner (called the subject) and the signing → [Certification Authority](#) (called the issuer), plus the owner's → [public key](#) and the signature made by the CA. Network entities verify these signatures using CA certificates.

See: [SSL/TLS Encryption](#)

## Certificate Signing Request (CSR)

An unsigned → [certificate](#) for submission to a → [Certification Authority](#), which signs it with the → [Private Key](#) of their CA *Certificate*. Once the CSR is signed, it becomes a real certificate.

See: [SSL/TLS Encryption](#)

## **Certification Authority (CA)**

A trusted third party whose purpose is to sign certificates for network entities it has authenticated using secure means. Other network entities can check the signature to verify that a CA has authenticated the bearer of a certificate.

See: [SSL/TLS Encryption](#)

## **Cipher**

An algorithm or system for data encryption. Examples are DES, IDEA, RC4, etc.

See: [SSL/TLS Encryption](#)

## **Ciphertext**

The result after → [Plaintext](#) is passed through a → [Cipher](#).

See: [SSL/TLS Encryption](#)

## **Common Gateway Interface (CGI)**

A standard definition for an interface between a web server and an external program that allows the external program to service requests. The interface was originally defined by [NCSA](#) but there is also an [RFC project](#).

See: [Dynamic Content with CGI](#)

## **Configuration Directive**

See: → [Directive](#)

## **Configuration File**

A text file containing → [Directives](#) that control the configuration of Apache.

See: [Configuration Files](#)

## **CONNECT**

An HTTP → [method](#) for proxying raw data channels over HTTP. It can be used to encapsulate other protocols, such as the SSL protocol.

## **Context**

An area in the → [configuration files](#) where certain types of

→ [directives](#) are allowed.

See: [Terms Used to Describe Apache Directives](#)

## Digital Signature

An encrypted text block that validates a certificate or other file.

A → [Certification Authority](#) creates a signature by generating a hash of the *Public Key* embedded in a *Certificate*, then encrypting the hash with its own *Private Key*. Only the CA's public key can decrypt the signature, verifying that the CA has authenticated the network entity that owns the *Certificate*.

See: [SSL/TLS Encryption](#)

## Directive

A configuration command that controls one or more aspects of Apache's behavior. Directives are placed in the

→ [Configuration File](#)

See: [Directive Index](#)

## Dynamic Shared Object (DSO)

→ [Modules](#) compiled separately from the Apache [httpd](#) binary that can be loaded on-demand.

See: [Dynamic Shared Object Support](#)

## Environment Variable (env-variable)

Named variables managed by the operating system shell and used to store information and communicate between programs. Apache also contains internal variables that are referred to as environment variables, but are stored in internal Apache structures, rather than in the shell environment.

See: [Environment Variables in Apache](#)

## Export-Crippled

Diminished in cryptographic strength (and security) in order to comply with the United States' Export Administration Regulations (EAR). Export-crippled cryptographic software is limited to a small key size, resulting in *Ciphertext* which usually can be decrypted by brute force.

See: [SSL/TLS Encryption](#)

## **Filter**

A process that is applied to data that is sent or received by the server. Input filters process data sent by the client to the server, while output filters process documents on the server before they are sent to the client. For example, the INCLUDES output filter processes documents for → [Server Side Includes](#).

See: [Filters](#)

## **Fully-Qualified Domain-Name (FQDN)**

The unique name of a network entity, consisting of a hostname and a domain name that can resolve to an IP address. For example, www is a hostname, example.com is a domain name, and www.example.com is a fully-qualified domain name.

## **Handler**

An internal Apache representation of the action to be performed when a file is called. Generally, files have implicit handlers, based on the file type. Normally, all files are simply served by the server, but certain file types are "handled" separately. For example, the cgi-script handler designates files to be processed as → [CGIs](#).

See: [Apache's Handler Use](#)

## **Hash**

A mathematical one-way, irreversable algorithm generating a string with fixed-length from another string of any length.

Different input strings will usually produce different hashes (depending on the hash function).

## **Header**

The part of the → [HTTP](#) request and response that is sent before the actual content, and that contains meta-information describing the content.

## **.htaccess**

A → [configuration file](#) that is placed inside the web tree and applies configuration → [directives](#) to the directory where it is placed and all sub-directories. Despite its name, this file can hold almost any type of directive, not just access-control directives.

See: [Configuration Files](#)

### **httpd.conf**

The main Apache → [configuration file](#). The default location is `/usr/local/apache2/conf/httpd.conf`, but it may be moved using run-time or compile-time configuration.

See: [Configuration Files](#)

### **HyperText Transfer Protocol (HTTP)**

The standard transmission protocol used on the World Wide Web. Apache implements version 1.1 of the protocol, referred to as HTTP/1.1 and defined by [RFC 2616](#).

### **HTTPS**

The HyperText Transfer Protocol (Secure), the standard encrypted communication mechanism on the World Wide Web. This is actually just HTTP over → [SSL](#).

See: [SSL/TLS Encryption](#)

### **Method**

In the context of → [HTTP](#), an action to perform on a resource, specified on the request line by the client. Some of the methods available in HTTP are GET, POST, and PUT.

### **Message Digest**

A hash of a message, which can be used to verify that the contents of the message have not been altered in transit.

See: [SSL/TLS Encryption](#)

### **MIME-type**

A way to describe the kind of document being transmitted. Its name comes from that fact that its format is borrowed from the Multipurpose Internet Mail Extensions. It consists of a

major type and a minor type, separated by a slash. Some examples are `text/html`, `image/gif`, and `application/octet-stream`. In HTTP, the MIME-type is transmitted in the Content-Type → [header](#).

See: [mod\\_mime](#)

## Module

An independent part of a program. Much of Apache's functionality is contained in modules that you can choose to include or exclude. Modules that are compiled into the Apache [httpd](#) binary are called *static modules*, while modules that are stored separately and can be optionally loaded at run-time are called *dynamic modules* or → [DSOs](#). Modules that are included by default are called *base modules*. Many modules are available for Apache that are not distributed as part of the Apache HTTP Server → [tarball](#). These are referred to as *third-party modules*.

See: [Module Index](#)

## Module Magic Number (MMN)

Module Magic Number is a constant defined in the Apache source code that is associated with binary compatibility of modules. It is changed when internal Apache structures, function calls and other significant parts of API change in such a way that binary compatibility cannot be guaranteed any more. On MMN change, all third party modules have to be at least recompiled, sometimes even slightly changed in order to work with the new version of Apache.

## OpenSSL

The Open Source toolkit for SSL/TLS

See <http://www.openssl.org/#>

## Pass Phrase

The word or phrase that protects private key files. It prevents unauthorized users from encrypting them. Usually it's just the

secret encryption/decryption key used for → [Ciphers](#).  
See: [SSL/TLS Encryption](#)

## **Plaintext**

The unencrypted text.

## **Private Key**

The secret key in a → [Public Key Cryptography](#) system, used to decrypt incoming messages and sign outgoing ones.

See: [SSL/TLS Encryption](#)

## **Proxy**

An intermediate server that sits between the client and the *origin server*. It accepts requests from clients, transmits those requests on to the origin server, and then returns the response from the origin server to the client. If several clients request the same content, the proxy can deliver that content from its cache, rather than requesting it from the origin server each time, thereby reducing response time.

See: [mod\\_proxy](#)

## **Public Key**

The publicly available key in a → [Public Key Cryptography](#) system, used to encrypt messages bound for its owner and to decrypt signatures made by its owner.

See: [SSL/TLS Encryption](#)

## **Public Key Cryptography**

The study and application of asymmetric encryption systems, which use one key for encryption and another for decryption. A corresponding pair of such keys constitutes a key pair. Also called Asymmetric Cryptography.

See: [SSL/TLS Encryption](#)

## **Regular Expression (Regex)**

A way of describing a pattern in text - for example, "all the words that begin with the letter A" or "every 10-digit phone number" or even "Every sentence with two commas in it, and

no capital letter Q". Regular expressions are useful in Apache because they let you apply certain attributes against collections of files or resources in very flexible ways - for example, all .gif and .jpg files under any "images" directory could be written as `"/images/.*(jpg|gif)$"`. Apache uses Perl Compatible Regular Expressions provided by the [PCRE](#) library.

### **Reverse Proxy**

A → [proxy](#) server that appears to the client as if it is an *origin server*. This is useful to hide the real origin server from the client for security reasons, or to load balance.

### **Secure Sockets Layer (SSL)**

A protocol created by Netscape Communications Corporation for general communication authentication and encryption over TCP/IP networks. The most popular usage is *HTTPS*, i.e. the HyperText Transfer Protocol (HTTP) over SSL.

See: [SSL/TLS Encryption](#)

### **Server Side Includes (SSI)**

A technique for embedding processing directives inside HTML files.

See: [Introduction to Server Side Includes](#)

### **Session**

The context information of a communication in general.

### **SSLey**

The original SSL/TLS implementation library developed by Eric A. Young

### **Symmetric Cryptography**

The study and application of *Ciphers* that use a single secret key for both encryption and decryption operations.

See: [SSL/TLS Encryption](#)

### **Tarball**

A package of files gathered together using the tar utility. Apache distributions are stored in compressed tar archives or using pkzip.

### **Transport Layer Security (TLS)**

The successor protocol to SSL, created by the Internet Engineering Task Force (IETF) for general communication authentication and encryption over TCP/IP networks. TLS version 1 is nearly identical with SSL version 3.

See: [SSL/TLS Encryption](#)

### **Uniform Resource Locator (URL)**

The name/address of a resource on the Internet. This is the common informal term for what is formally called a → [Uniform Resource Identifier](#). URLs are usually made up of a scheme, like http or https, a hostname, and a path. A URL for this page is

<http://httpd.apache.org/docs/2.0/glossary.html>

### **Uniform Resource Identifier (URI)**

A compact string of characters for identifying an abstract or physical resource. It is formally defined by [RFC 2396](#). URIs used on the world-wide web are commonly referred to as → [URLs](#).

### **Virtual Hosting**

Serving multiple websites using a single instance of Apache. *IP virtual hosting* differentiates between websites based on their IP address, while *name-based virtual hosting* uses only the name of the host and can therefore host many sites on the same IP address.

See: [Apache Virtual Host documentation](#)

### **X.509**

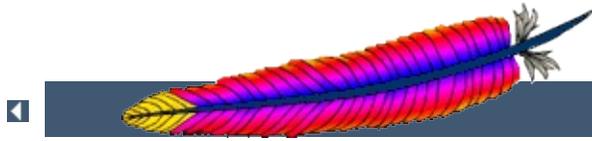
An authentication certificate scheme recommended by the International Telecommunication Union (ITU-T) which is used for SSL/TLS authentication.

See: [SSL/TLS Encryption](#)

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >

Apache Apache

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](#)

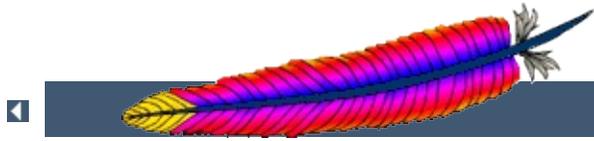
- [MaxClients](#)
- [MaxKeepAliveRequests](#)
- [MaxMemFree](#)
- [MaxRanges](#)
- [MaxRequestsPerChild](#)
- [MaxRequestsPerThread](#)
- [MaxSpareServers](#)
- [MaxSpareThreads](#)
- [MaxThreads](#)
- [MaxThreadsPerChild](#)
- [MCacheMaxObjectCount](#)
- [MCacheMaxObjectSize](#)
- [MCacheMaxStreamingBuffer](#)
- [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 2.0

[Apache](#) > [HTTP](#) > > [2.0](#) >



This translation may be out of date. Check the English version for recent changes.

## Apache

1

2 ()

3, 4

A	B	C	D	E	F	G	H
I	K	L	M	N	O	P	R
S	T	U	V	W	X		

s	<b>C</b> Core
v	<b>M</b> MPM
d	<b>B</b> Base
h	<b>E</b> Extension
.htaccess	<b>X</b> Experimental

<a href="#">AcceptMutex default method</a> accept Apache	default
<a href="#">AcceptPathInfo On Off Default</a>	Default
<a href="#">AccessFileName filename [filename] ...</a>	.htaccess
<a href="#">Action action-type cgi-script</a> CGI	
<a href="#">AddAlt string file [file] ...</a>	
<a href="#">AddAltByEncoding string MIME-encoding [MIME-encoding] ...</a> MIME	
<a href="#">AddAltByType string MIME-type [MIME-type] ...</a> MIME	
<a href="#">AddCharset charset extension [extension] ...</a>	

<a href="#">AddDefaultCharset On Off charset</a>	Off
<a href="#">AddDescription string file [file] ...</a>	
<a href="#">AddEncoding MIME-enc extension [extension] ...</a>	
<a href="#">AddHandler handler-name extension [extension] ...</a>	
<a href="#">AddIcon icon name [name] ...</a>	
<a href="#">AddIconByEncoding icon MIME-encoding [MIME-encoding] ...</a> MIME	
<a href="#">AddIconByType icon MIME-type [MIME-type] ...</a> MIME	
<a href="#">AddInputFilter filter[;filter...] extension [extension] ...</a>	
<a href="#">AddLanguage MIME-lang extension [extension] ...</a>	
<a href="#">AddModuleInfo module-name string</a> server-info	
<a href="#">AddOutputFilter filter[;filter...] extension [extension] ...</a>	
<a href="#">AddOutputFilterByType filter[;filter...] MIME-type [MIME-type] ...</a> MIME-type	
<a href="#">AddType MIME-type extension [extension] ...</a>	

<a href="#">Alias</a> <i>URL-path file-path directory-path</i>	
URL	
<a href="#">AliasMatch</a> <i>regex file-path directory-path</i>	
URL	
<a href="#">Allow from</a> <i>all host env=env-variable</i> <i>[host env=env-variable] ...</i>	
<a href="#">AllowCONNECT</a> <i>port [port] ...</i>	443 563
Ports that are allowed to CONNECT through the proxy	
<a href="#">AllowEncodedSlashes</a> <i>On Off</i>	Off
URL	
<a href="#">AllowOverride</a> <i>All None directive-type</i> <i>[directive-type] ...</i>	All
.htaccess	
<a href="#">Anonymous</a> <i>user [user] ...</i>	
Specifies userIDs that are allowed access without password verification	
<a href="#">Anonymous_Authoritative</a> <i>On Off</i>	Off
Configures if authorization will fall-through to other methods	
<a href="#">Anonymous_LogEmail</a> <i>On Off</i>	On
Sets whether the password entered will be logged in the error log	
<a href="#">Anonymous_MustGiveEmail</a> <i>On Off</i>	On
Specifies whether blank passwords are allowed	
<a href="#">Anonymous_NoUserID</a> <i>On Off</i>	Off
Sets whether the userID field may be empty	
<a href="#">Anonymous_VerifyEmail</a> <i>On Off</i>	Off
Sets whether to check the password field for a correctly formatted email address	
<a href="#">AssignUserID</a> <i>user-id group-id</i>	
Tie a virtual host to a user and group ID	
<a href="#">AuthAuthoritative</a> <i>On Off</i>	On
<a href="#">AuthDBMAuthoritative</a> <i>On Off</i>	On
Sets whether authentication and authorization will be passed on to lower level modules	
<a href="#">AuthDBMGroupFile</a> <i>file-path</i>	
Sets the name of the database file containing the list of user groups for authentication	
<a href="#">AuthDBMType</a>	default

<a href="#">default SDBM GDBM NDBM DB</a>	
Sets the type of database file that is used to store passwords	
<a href="#">AuthDBMUserFile <i>file-path</i></a>	
Sets the name of a database file containing the list of users and passwords for authentication	
<a href="#">AuthDigestAlgorithm MD5 MD5-sess</a>	MD5
Selects the algorithm used to calculate the challenge and response hashes in digest authentication	
<a href="#">AuthDigestDomain <i>URI [URI] ...</i></a>	
URIs that are in the same protection space for digest authentication	
<a href="#">AuthDigestFile <i>file-path</i></a>	
Location of the text file containing the list of users and encoded passwords for digest authentication	
<a href="#">AuthDigestGroupFile <i>file-path</i></a>	
Name of the text file containing the list of groups for digest authentication	
<a href="#">AuthDigestNcCheck On Off</a>	Off
Enables or disables checking of the nonce-count sent by the server	
<a href="#">AuthDigestNonceFormat <i>format</i></a>	
Determines how the nonce is generated	
<a href="#">AuthDigestNonceLifetime <i>seconds</i></a>	300
How long the server nonce is valid	
<a href="#">AuthDigestQop none auth auth-int [auth auth-int]</a>	auth
Determines the quality-of-protection to use in digest authentication	
<a href="#">AuthDigestShmemSize <i>size</i></a>	1000
The amount of shared memory to allocate for keeping track of clients	
<a href="#">AuthGroupFile <i>file-path</i></a>	
<a href="#">AuthLDAPAuthoritative on off</a>	on
Prevent other authentication modules from authenticating the user if this one fails	
<a href="#">AuthLDAPBindDN <i>distinguished-name</i></a>	
Optional DN to use in binding to the LDAP server	
<a href="#">AuthLDAPBindPassword <i>password</i></a>	
Password used in conjunction with the bind DN	
<a href="#">AuthLDAPCharsetConfig <i>file-path</i></a>	
Language to charset conversion configuration file	
<a href="#">AuthLDAPCompareDNOnServer on off</a>	on
Use the LDAP server to compare the DNS	

<a href="#">AuthLDAPDereferenceAliases</a> <a href="#">never searching finding always</a> When will the module de-reference aliases	Always
<a href="#">AuthLDAPEnabled</a> <a href="#">on off</a> Turn on or off LDAP authentication	on
<a href="#">AuthLDAPFrontPageHack</a> <a href="#">on off</a> Allow LDAP authentication to work with MS FrontPage	off
<a href="#">AuthLDAPGroupAttribute</a> <i>attribute</i> LDAP attributes used to check for group membership	
<a href="#">AuthLDAPGroupAttributeIsDN</a> <a href="#">on off</a> Use the DN of the client username when checking for group membership	on
<a href="#">AuthLDAPRemoteUserIsDN</a> <a href="#">on off</a> Use the DN of the client username to set the REMOTE_USER environment variable	off
<a href="#">AuthLDAPUrl</a> <i>url</i> URL specifying the LDAP search parameters	
<a href="#">AuthName</a> <i>auth-domain</i> HTTP (: realm)	
<a href="#">AuthType</a> <a href="#">Basic Digest</a>	
<a href="#">AuthUserFile</a> <i>file-path</i>	
<a href="#">BrowserMatch</a> <i>regex [!]<a href="#">env-variable</a>[=<a href="#">value</a>] [[!]<a href="#">env-variable</a>[=<a href="#">value</a>]] ...</i> HTTP User-Agent	
<a href="#">BrowserMatchNoCase</a> <i>regex [!]<a href="#">env-variable</a>[=<a href="#">value</a>] [[!]<a href="#">env-variable</a>[=<a href="#">value</a>]] ...</i> HTTP User-Agent	
<a href="#">BS2000Account</a> <i>account</i> BS2000	
<a href="#">BufferedLogs</a> <a href="#">On Off</a> Buffer log entries in memory before writing to disk	Off
<a href="#">CacheDefaultExpire</a> <i>seconds</i> The default duration to cache a document when no expiry date is specified.	3600 (one hour)
<a href="#">CacheDirLength</a> <i>length</i> The number of characters in subdirectory names	2

<a href="#"><u>CacheDirLevels</u></a> <i>levels</i>	3
The number of levels of subdirectories in the cache.	
<a href="#"><u>CacheDisable</u></a> <i>url-string</i>	
Disable caching of specified URLs	
<a href="#"><u>CacheEnable</u></a> <i>cache_type url-string</i>	
Enable caching of specified URLs using a specified storage manager	
<a href="#"><u>CacheExpiryCheck</u></a> On Off	On
Indicates if the cache observes Expires dates when seeking files	
<a href="#"><u>CacheFile</u></a> <i>file-path [file-path] ...</i>	
Cache a list of file handles at startup time	
<a href="#"><u>CacheForceCompletion</u></a> <i>Percentage</i>	60
Percentage of document served, after which the server will complete caching the file even if cancelled.	
<a href="#"><u>CacheGcClean</u></a> <i>hours url-string</i>	?
The time to retain unchanged cached files that match a URL	
<a href="#"><u>CacheGcDaily</u></a> <i>time</i>	?
The recurring time each day for garbage collection to be run. (24 hour clock)	
<a href="#"><u>CacheGcInterval</u></a> <i>hours</i>	
The interval between garbage collection attempts.	
<a href="#"><u>CacheGcMemUsage</u></a> <i>KBytes</i>	?
The maximum kilobytes of memory used for garbage collection	
<a href="#"><u>CacheGcUnused</u></a> <i>hours url-string</i>	?
The time to retain unreferenced cached files that match a URL.	
<a href="#"><u>CacheIgnoreCacheControl</u></a> On Off	Off
Ignore the fact that the client requested the content not be cached.	
<a href="#"><u>CacheIgnoreHeaders</u></a> <i>header-string [header-string] ...</i>	None
Do not store the given HTTP header(s) in the cache.	
<a href="#"><u>CacheIgnoreNoLastMod</u></a> On Off	Off
Ignore the fact that a response has no Last Modified header.	
<a href="#"><u>CacheLastModifiedFactor</u></a> <i>float</i>	0.1
The factor used to compute an expiry date based on the LastModified date.	
<a href="#"><u>CacheMaxExpire</u></a> <i>seconds</i>	86400 (one day)
The maximum time in seconds to cache a document	
<a href="#"><u>CacheMaxFileSize</u></a> <i>bytes</i>	1000000

<a href="#">CacheMinFileSize</a> <i>bytes</i>	1
The maximum size (in bytes) of a document to be placed in the cache	
<a href="#">CacheNegotiatedDocs</a> On Off	Off
The minimum size (in bytes) of a document to be placed in the cache	
<a href="#">CacheRoot</a> <i>directory</i>	
The directory root under which cache files are stored	
<a href="#">CacheSize</a> <i>KBytes</i>	1000000
The maximum amount of disk space that will be used by the cache in KBytes	
<a href="#">CacheTimeMargin</a> ?	?
The minimum time margin to cache a document	
<a href="#">CGIMapExtension</a> <i>cgi-path .extension</i>	
CGI	
<a href="#">CharsetDefault</a> <i>charset</i>	
Charset to translate into	
<a href="#">CharsetOptions</a> <i>option [option] ...</i>	DebugLevel=0 NoImp
Configures charset translation behavior	
<a href="#">CharsetSourceEnc</a> <i>charset</i>	
Source charset of files	
<a href="#">CheckSpelling</a> on off	Off
spelling	
<a href="#">ChildPerUserID</a> <i>user-id group-id num-children</i>	
Specify user ID and group ID for a number of child processes	
<a href="#">ContentDigest</a> On Off	Off
Content -MD5 HTTP	
<a href="#">CookieDomain</a> <i>domain</i>	
The domain to which the tracking cookie applies	
<a href="#">CookieExpires</a> <i>expiry-period</i>	
Expiry time for the tracking cookie	
<a href="#">CookieLog</a> <i>filename</i>	
<a href="#">CookieName</a> <i>token</i>	Apache
Name of the tracking cookie	
<a href="#">CookieStyle</a>	Netscape
<a href="#">Netscape Cookie Cookie2 RFC2109 RFC2965</a>	

Format of the cookie header field	
<a href="#">CookieTracking on off</a>	off
Enables tracking cookie	
<a href="#">CoreDumpDirectory <i>directory</i></a>	
Apache	
<a href="#">CustomLog <i>file pipe format nickname</i> [env= [!]<i>environment-variable</i>]</a>	
<a href="#">Dav On Off <i>provider-name</i></a>	Off
WebDAV HTTP	
<a href="#">DavDepthInfinity on off</a>	off
PROPFIND, Depth: Infinity	
<a href="#">DavLockDB <i>file-path</i></a>	
DAV	
<a href="#">DavMinTimeout <i>seconds</i></a>	0
DAV	
<a href="#">DefaultIcon <i>url-path</i></a>	
<a href="#">DefaultLanguage <i>MIME-lang</i></a>	
<a href="#">DefaultType <i>MIME-type</i></a>	text/plain
MIME	
<a href="#">DeflateBufferSize <i>value</i></a>	8096
zlib	
<a href="#">DeflateCompressionLevel <i>value</i></a>	
<a href="#">DeflateFilterNote [<i>type</i>] <i>notename</i></a>	
<a href="#">DeflateMemLevel <i>value</i></a>	9
zlib	
<a href="#">DeflateWindowSize <i>value</i></a>	15
Zlib	
<a href="#">Deny from all <i>host</i>[env=<i>env-variable</i>] [<i>host</i>[env=<i>env-variable</i>] ...</a>	

<a href="#">&lt;Directory <i>directory-path</i>&gt; ... &lt;/Directory&gt;</a>	
<a href="#">DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...</a>	index.html
<a href="#">&lt;DirectoryMatch <i>regex</i>&gt; ... &lt;/DirectoryMatch&gt;</a>	
<a href="#">DirectorySlash On Off</a>	On
<a href="#">DocumentRoot <i>directory-path</i></a>	/usr/local/apache/h +
<a href="#">DumpIOInput On Off</a> Dump all input data to the error log	Off
<a href="#">DumpIOOutput On Off</a> Dump all output data to the error log	Off
<a href="#">EnableExceptionHook On Off</a>	Off
<a href="#">EnableMMAP On Off</a>	On
<a href="#">EnableSendfile On Off</a> sendfile	On
<a href="#">ErrorDocument <i>error-code</i> <i>document</i></a>	
<a href="#">ErrorLog <i>file-path</i> syslog[:<i>facility</i>]</a>	logs/error_log (Uni +
<a href="#">Example</a> Demonstration directive to illustrate the Apache module API	
<a href="#">ExpiresActive On Off</a> Expires	
<a href="#">ExpiresByType <i>MIME-type</i> <code>&lt;code&gt;seconds</code></a> MIME Expires	
<a href="#">ExpiresDefault <code>&lt;code&gt;seconds</code></a>	
<a href="#">ExtendedStatus On Off</a>	Off
<a href="#">ExtFilterDefine <i>filtername</i> <i>parameters</i></a>	

Define an external filter	
<a href="#">ExtFilterOptions <i>option</i> [<i>option</i>] ...</a>	DebugLevel=0 NoLog +
Configure <code>mod_ext_filter</code> options	
<a href="#">FileETag <i>component</i> ...</a>	Inode MTime Size
ETag HTTP	
<a href="#">&lt;Files <i>filename</i>&gt; ... &lt;/Files&gt;</a>	
<a href="#">&lt;FilesMatch <i>regex</i>&gt; ... &lt;/FilesMatch&gt;</a>	
<a href="#">ForceLanguagePriority None Prefer Fallback [Prefer Fallback]</a>	Prefer
<a href="#">ForceType <i>MIME-type</i> None</a>	
MIME	
<a href="#">ForensicLog <i>filename</i> <i>pipe</i></a>	
Sets filename of the forensic log	
<a href="#">Group <i>unix-group</i></a>	#-1
<a href="#">Header [<i>condition</i>] set append add unset echo <i>header</i> [<i>value</i>] [<i>env</i>=[!]<i>variable</i>]</a>	
Configure HTTP response headers	
<a href="#">HeaderName <i>filename</i></a>	
<a href="#">HostnameLookups On Off Double</a>	Off
IP DNS	
<a href="#">IdentityCheck On Off</a>	Off
RFC1413	
<a href="#">&lt;IfDefine [!]<i>parameter-name</i>&gt; ... &lt;/IfDefine&gt;</a>	
<a href="#">&lt;IfModule [!]<i>module-name</i>&gt; ... &lt;/IfModule&gt;</a>	
<a href="#">&lt;IfVersion [!]<i>operator</i> <i>version</i>&gt; ... &lt;/IfVersion&gt;</a>	

<a href="#">ImapBase</a> <a href="#">map referer URL</a> Default base for imagemap files	http://servername/
<a href="#">ImapDefault</a> <a href="#">error nocontent map referer URL</a> Default action when an imagemap is called with coordinates that are not explicitly mapped	nocontent
<a href="#">ImapMenu</a> <a href="#">none formatted semiformatted unformatted</a> Action if no coordinates are given when calling an imagemap	
<a href="#">Include</a> <a href="#">file-path directory-path</a>	
<a href="#">IndexIgnore</a> <a href="#">file [file] ...</a>	
<a href="#">IndexOptions</a> <a href="#">[+ -]option [[+ -]option] ...</a>	
<a href="#">IndexOrderDefault</a> <a href="#">Ascending Descending</a> <a href="#">Name Date Size Description</a>	Ascending Name
<a href="#">ISAPIAppendLogToErrors</a> <a href="#">on off</a> Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log	off
<a href="#">ISAPIAppendLogToQuery</a> <a href="#">on off</a> Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field	on
<a href="#">ISAPICacheFile</a> <a href="#">file-path [file-path] ...</a> ISAPI .dll files to be loaded at startup	
<a href="#">ISAPIFakeAsync</a> <a href="#">on off</a> Fake asynchronous support for ISAPI callbacks	off
<a href="#">ISAPILogNotSupported</a> <a href="#">on off</a> Log unsupported feature requests from ISAPI extensions	off
<a href="#">ISAPIReadAheadBuffer</a> <a href="#">size</a> Size of the Read Ahead Buffer sent to ISAPI extensions	49152
<a href="#">KeepAlive</a> <a href="#">On Off</a> HTTP	On
<a href="#">KeepAliveTimeout</a> <a href="#">seconds</a>	15
<a href="#">LanguagePriority</a> <a href="#">MIME-lang [MIME-lang] ...</a> variant	
<a href="#">LDAPCacheEntries</a> <a href="#">number</a>	1024

Maximum number of entries in the primary LDAP cache	
<a href="#"><u>LDAPCacheTTL seconds</u></a>	600
Time that cached items remain valid	
<a href="#"><u>LDAPConnectionTimeout seconds</u></a>	
Specifies the socket connection timeout in seconds	
<a href="#"><u>LDAPOpCacheEntries number</u></a>	1024
Number of entries used to cache LDAP compare operations	
<a href="#"><u>LDAPOpCacheTTL seconds</u></a>	600
Time that entries in the operation cache remain valid	
<a href="#"><u>LDAPSharedCacheFile directory-path/filename</u></a>	
Sets the shared memory cache file	
<a href="#"><u>LDAPSharedCacheSize bytes</u></a>	102400
Size in bytes of the shared-memory cache	
<a href="#"><u>LDAPTrustedCA directory-path/filename</u></a>	
Sets the file containing the trusted Certificate Authority certificate or database	
<a href="#"><u>LDAPTrustedCAType type</u></a>	
Specifies the type of the Certificate Authority file	
<a href="#"><u>&lt;Limit method [method] ... &gt; ... &lt;/Limit&gt;</u></a>	
HTTP	
<a href="#"><u>&lt;LimitExcept method [method] ... &gt; ... &lt;/LimitExcept&gt;</u></a>	
HTTP	
<a href="#"><u>LimitInternalRecursion number [number]</u></a>	10
<a href="#"><u>LimitRequestBody bytes</u></a>	0
HTTP	
<a href="#"><u>LimitRequestFields number</u></a>	100
HTTP	
<a href="#"><u>LimitRequestFieldsize bytes</u></a>	
HTTP	
<a href="#"><u>LimitRequestLine bytes</u></a>	8190
HTTP	
<a href="#"><u>LimitXMLRequestBody bytes</u></a>	1000000
XML	

<a href="#">Listen</a> <i>[IP-address:]portnumber</i>	
listen IP	
<a href="#">ListenBacklog</a> <i>backlog</i>	
<a href="#">LoadFile</a> <i>filename [filename] ...</i>	
<a href="#">LoadModule</a> <i>module filename</i>	
<a href="#">&lt;Location</a> <i>URL-path URL&gt; ... &lt;/Location&gt;</i>	
URL	
<a href="#">&lt;LocationMatch</a> <i>regex&gt; ... &lt;/LocationMatch&gt;</i>	
URL	
<a href="#">LockFile</a> <i>filename</i>	logs/accept.lock
<a href="#">LogFormat</a> <i>format nickname [nickname]</i>	"%h %l %u %t \"%r\" -
<a href="#">LogLevel</a> <i>level</i>	warn
ErrorLog	
<a href="#">MaxClients</a> <i>number</i>	
<a href="#">MaxKeepAliveRequests</a> <i>number</i>	100
<a href="#">MaxMemFree</a> <i>KBytes</i>	0
free()	
<a href="#">MaxRanges</a> <i>default   unlimited   none   number-of-ranges</i>	200
Number of ranges allowed before returning the complete resource	
<a href="#">MaxRequestsPerChild</a> <i>number</i>	10000
<a href="#">MaxRequestsPerThread</a> <i>number</i>	0
Limit on the number of requests that an individual thread will handle during its life	
<a href="#">MaxSpareServers</a> <i>number</i>	10
<a href="#">MaxSpareThreads</a> <i>number</i>	

<a href="#"><u>MaxThreads</u></a> <i>number</i>	2048
Set the maximum number of worker threads	
<a href="#"><u>MaxThreadsPerChild</u></a> <i>number</i>	64
Maximum number of threads per child process	
<a href="#"><u>MCacheMaxObjectCount</u></a> <i>value</i>	1009
<a href="#"><u>MCacheMaxObjectSize</u></a> <i>bytes</i>	10000
()	
<a href="#"><u>MCacheMaxStreamingBuffer</u></a> <i>size in bytes</i>	of 100000 MCacheM +
<a href="#"><u>MCacheMinObjectSize</u></a> <i>bytes</i>	0
()	
<a href="#"><u>MCacheRemovalAlgorithm</u></a> LRU GDSF	GDSF
<a href="#"><u>MCacheSize</u></a> <i>KBytes</i>	100
<a href="#"><u>MetaDir</u></a> <i>directory</i>	.web
Name of the directory to find CERN-style meta information files	
<a href="#"><u>MetaFiles</u></a> on off	off
Activates CERN meta-file processing	
<a href="#"><u>MetaSuffix</u></a> <i>suffix</i>	.meta
File name suffix for the file containing CERN-style meta information	
<a href="#"><u>MimeMagicFile</u></a> <i>file-path</i>	
Enable MIME-type determination based on file contents using the specified magic file	
<a href="#"><u>MinSpareServers</u></a> <i>number</i>	5
<a href="#"><u>MinSpareThreads</u></a> <i>number</i>	
<a href="#"><u>MMapFile</u></a> <i>file-path [file-path] ...</i>	
Map a list of files into memory at startup time	
<a href="#"><u>ModMimeUsePathInfo</u></a> On Off	Off
path_info <a href="#"><u>mod_mime</u></a>	
<a href="#"><u>MultiviewsMatch</u></a>	NegotiatedOnly
<a href="#"><u>Any NegotiatedOnly Filters Handlers</u></a>	

<a href="#">[Handlers Filters]</a>	
MultiViews	
<a href="#">NameVirtualHost <i>addr[:port]</i></a>	
IP	
<a href="#">NoProxy <i>host [host] ...</i></a>	
Hosts, domains, or networks that will be connected to directly	
<a href="#">NumServers <i>number</i></a>	2
Total number of children alive at the same time	
<a href="#">NWSSLTrustedCerts <i>filename [filename] ...</i></a>	
List of additional client certificates	
<a href="#">NWSSLUpgradeable [<i>IP-address:]portnumber</i></a>	
Allows a connection to be upgraded to an SSL connection upon request	
<a href="#">Options [+ -]<i>option</i> [[+ -]<i>option</i>] ...</a>	All
<a href="#">Order <i>ordering</i></a>	Deny,Allow
Allow Deny	
<a href="#">PassEnv <i>env-variable [env-variable] ...</i></a>	
<a href="#">PidFile <i>filename</i></a>	logs/httpd.pid
ID	
<a href="#">ProtocolEcho On Off</a>	
<a href="#">&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</a>	
Container for directives applied to proxied resources	
<a href="#">ProxyBadHeader IsError Ignore StartBody</a>	IsError
Determines how to handle bad header lines in a response	
<a href="#">ProxyBlock * <i>word</i> <i>host</i> <i>domain</i> [<i>word</i> <i>host</i> <i>domain</i>] ...</a>	
Words, hosts, or domains that are banned from being proxied	
<a href="#">ProxyDomain <i>Domain</i></a>	
Default domain name for proxied requests	
<a href="#">ProxyErrorOverride On Off</a>	Off
Override error pages for proxied content	
<a href="#">ProxyFtpDirCharset <i>character set</i></a>	ISO-8859-1
Define the character set for proxied FTP listings	

<a href="#">ProxyIOBufferSize <i>bytes</i></a>	8192
Determine size of internal data throughput buffer	
<a href="#">&lt;ProxyMatch <i>regex</i>&gt; ...&lt;/ProxyMatch&gt;</a>	
Container for directives applied to regular-expression-matched proxied resources	
<a href="#">ProxyMaxForwards <i>number</i></a>	10
Maximum number of proxies that a request can be forwarded through	
<a href="#">ProxyPass [<i>path</i>] <i>! url</i></a>	
Maps remote servers into the local server URL-space	
<a href="#">ProxyPassReverse [<i>path</i>] <i>url</i></a>	
Adjusts the URL in HTTP response headers sent from a reverse proxied server	
<a href="#">ProxyPreserveHost On Off</a>	Off
Use incoming Host HTTP request header for proxy request	
<a href="#">ProxyReceiveBufferSize <i>bytes</i></a>	0
Network buffer size for proxied HTTP and FTP connections	
<a href="#">ProxyRemote <i>match remote-server</i></a>	
Remote proxy used to handle certain requests	
<a href="#">ProxyRemoteMatch <i>regex remote-server</i></a>	
Remote proxy used to handle requests matched by regular expressions	
<a href="#">ProxyRequests On Off</a>	Off
Enables forward (standard) proxy requests	
<a href="#">ProxyTimeout <i>seconds</i></a>	300
Network timeout for proxied requests	
<a href="#">ProxyVia On Off Full Block</a>	Off
Information provided in the Via HTTP response header for proxied requests	
<a href="#">ReadmeName <i>filename</i></a>	
<a href="#">ReceiveBufferSize <i>bytes</i></a>	0
TCP receive buffer size	
<a href="#">Redirect [<i>status</i>] <i>URL-path URL</i></a>	
URL	
<a href="#">RedirectMatch [<i>status</i>] <i>regex URL</i></a>	
URL	
<a href="#">RedirectPermanent <i>URL-path URL</i></a>	
URL	
<a href="#">RedirectTemp <i>URL-path URL</i></a>	

URL	
<a href="#">RemoveCharset <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RemoveEncoding <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RemoveHandler <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RemoveInputFilter <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RemoveLanguage <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RemoveOutputFilter <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RemoveType <i>extension</i> [<i>extension</i>] ...</a>	
<a href="#">RequestHeader <i>set append add unset header</i> [<i>value</i> [<i>env</i>=[!]<i>variable</i>]]</a>	
Configure HTTP request headers	
<a href="#">Require <i>entity-name</i> [<i>entity-name</i>] ...</a>	
<a href="#">RewriteBase <i>URL-path</i></a>	
Sets the base URL for per-directory rewrites	
<a href="#">RewriteCond <i>TestString CondPattern</i></a>	
Defines a condition under which rewriting will take place	
<a href="#">RewriteEngine <i>on off</i></a>	off
Enables or disables runtime rewriting engine	
<a href="#">RewriteLock <i>file-path</i></a>	
Sets the name of the lock file used for <a href="#">RewriteMap</a> synchronization	
<a href="#">RewriteLog <i>file-path</i></a>	
Sets the name of the file used for logging rewrite engine processing	
<a href="#">RewriteLogLevel <i>Level</i></a>	0
Sets the verbosity of the log file used by the rewrite engine	
<a href="#">RewriteMap <i>MapName MapType:MapSource</i></a>	
Defines a mapping function for key-lookup	
<a href="#">RewriteOptions <i>Options</i></a>	MaxRedirects=10

Sets some special options for the rewrite engine	
<a href="#">RewriteRule <i>Pattern Substitution</i></a>	
Defines rules for the rewriting engine	
<a href="#">RLimitCPU <i>seconds max [seconds max]</i></a>	
Apache CPU	
<a href="#">RLimitMEM <i>bytes max [bytes max]</i></a>	
Apache	
<a href="#">RLimitNPROC <i>number max [number max]</i></a>	
Apache	
<a href="#">Satisfy Any All</a>	All
<a href="#">ScoreBoardFile <i>file-path</i></a>	logs/apache_status
<a href="#">Script method <i>cgi-script</i></a>	
CGI	
<a href="#">ScriptAlias <i>URL-path file-path directory-path</i></a>	
URL CGI	
<a href="#">ScriptAliasMatch <i>regex file-path directory-path</i></a>	
URL CGI	
<a href="#">ScriptInterpreterSource Registry Registry-Strict Script</a>	Script
CGI	
<a href="#">ScriptLog <i>file-path</i></a>	
CGI	
<a href="#">ScriptLogBuffer <i>bytes</i></a>	1024
PUT POST	
<a href="#">ScriptLogLength <i>bytes</i></a>	10385760
CGI	
<a href="#">ScriptSock <i>file-path</i></a>	logs/cgisock
CGI	
<a href="#">SecureListen [<i>IP-address:</i>]portnumber <i>Certificate-Name [MUTUAL]</i></a>	
Enables SSL encryption for the specified port	
<a href="#">SendBufferSize <i>bytes</i></a>	0
TCP	

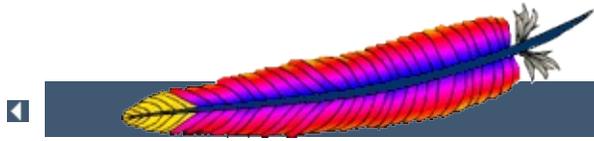
<a href="#"><u>ServerAdmin</u></a> <i>email-address</i>	
<a href="#"><u>ServerAlias</u></a> <i>hostname [hostname] ...</i>	
<a href="#"><u>ServerLimit</u></a> <i>number</i>	
<a href="#"><u>ServerName</u></a> <i>fully-qualified-domain-name[:port]</i>	
<a href="#"><u>ServerPath</u></a> <i>URL-path</i> URL	
<a href="#"><u>ServerRoot</u></a> <i>directory-path</i>	/usr/local/apache
<a href="#"><u>ServerSignature</u></a> On Off EMail	Off
<a href="#"><u>ServerTokens</u></a> Major Minor Min[imal] Prod[uctOnly] OS Full Server HTTP	Full
<a href="#"><u>SetEnv</u></a> <i>env-variable value</i>	
<a href="#"><u>SetEnvIf</u></a> <i>attribute regex [!]<u>env-variable</u>[=value] [[!]<u>env-variable</u>[=value]] ...</i>	
<a href="#"><u>SetEnvIfNoCase</u></a> <i>attribute regex [!]<u>env-variable</u>[=value] [[!]<u>env-variable</u>[=value]] ...</i>	
<a href="#"><u>SetHandler</u></a> <i>handler-name None</i>	
<a href="#"><u>SetInputFilter</u></a> <i>filter[:filter...]</i> POST	
<a href="#"><u>SetOutputFilter</u></a> <i>filter[:filter...]</i>	
<a href="#"><u>SSIEndTag</u></a> <i>tag</i> include	"-->"
<a href="#"><u>SSSErrorMsg</u></a> <i>message</i>	"[an error occurred +

SSI	
<a href="#"><u>SSIStartTag <i>tag</i></u></a>	"<!--#"
include	
<a href="#"><u>SSITimeFormat <i>formatstring</i></u></a>	"%A, %d-%b-%Y %H:%M +"
<a href="#"><u>SSIUndefinedEcho <i>string</i></u></a>	"(none)"
echo	
<a href="#"><u>SSLCACertificateFile <i>file-path</i></u></a>	
File of concatenated PEM-encoded CA Certificates for Client Auth	
<a href="#"><u>SSLCACertificatePath <i>directory-path</i></u></a>	
Directory of PEM-encoded CA Certificates for Client Auth	
<a href="#"><u>SSLCARevocationFile <i>file-path</i></u></a>	
File of concatenated PEM-encoded CA CRLs for Client Auth	
<a href="#"><u>SSLCARevocationPath <i>directory-path</i></u></a>	
Directory of PEM-encoded CA CRLs for Client Auth	
<a href="#"><u>SSLCertificateChainFile <i>file-path</i></u></a>	
File of PEM-encoded Server CA Certificates	
<a href="#"><u>SSLCertificateFile <i>file-path</i></u></a>	
Server PEM-encoded X.509 Certificate file	
<a href="#"><u>SSLCertificateKeyFile <i>file-path</i></u></a>	
Server PEM-encoded Private Key file	
<a href="#"><u>SSLCipherSuite <i>cipher-spec</i></u></a>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL handshake	
<a href="#"><u>SSLEngine <i>on off</i></u></a>	off
SSL Engine Operation Switch	
<a href="#"><u>SSLHonorCipherOrder <i>flag</i></u></a>	
Option to prefer the server's cipher preference order	
<a href="#"><u>SSLInsecureRenegotiation <i>flag</i></u></a>	off
Option to enable support for insecure renegotiation	
<a href="#"><u>SSLMutex <i>type</i></u></a>	none
Semaphore for internal mutual exclusion of operations	
<a href="#"><u>SSLOptions [<i>+</i> -]<i>option</i> ...</u></a>	
Configure various SSL engine run-time options	

<a href="#"><u>SSLPassPhraseDialog <i>type</i></u></a>	builtin
Type of pass phrase dialog for encrypted private keys	
<a href="#"><u>SSLProtocol <i>[+ -]protocol ...</i></u></a>	all
Configure usable SSL protocol flavors	
<a href="#"><u>SSLProxyCACertificateFile <i>file-path</i></u></a>	
File of concatenated PEM-encoded CA Certificates for Remote Server Auth	
<a href="#"><u>SSLProxyCACertificatePath <i>directory-path</i></u></a>	
Directory of PEM-encoded CA Certificates for Remote Server Auth	
<a href="#"><u>SSLProxyCARevocationFile <i>file-path</i></u></a>	
File of concatenated PEM-encoded CA CRLs for Remote Server Auth	
<a href="#"><u>SSLProxyCARevocationPath <i>directory-path</i></u></a>	
Directory of PEM-encoded CA CRLs for Remote Server Auth	
<a href="#"><u>SSLProxyCipherSuite <i>cipher-spec</i></u></a>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL proxy handshake	
<a href="#"><u>SSLProxyEngine <i>on off</i></u></a>	off
SSL Proxy Engine Operation Switch	
<a href="#"><u>SSLProxyMachineCertificateFile <i>filename</i></u></a>	
File of concatenated PEM-encoded client certificates and keys to be used by the proxy	
<a href="#"><u>SSLProxyMachineCertificatePath <i>directory</i></u></a>	
Directory of PEM-encoded client certificates and keys to be used by the proxy	
<a href="#"><u>SSLProxyProtocol <i>[+ -]protocol ...</i></u></a>	all
Configure usable SSL protocol flavors for proxy usage	
<a href="#"><u>SSLProxyVerify <i>level</i></u></a>	none
Type of remote server Certificate verification	
<a href="#"><u>SSLProxyVerifyDepth <i>number</i></u></a>	1
Maximum depth of CA Certificates in Remote Server Certificate verification	
<a href="#"><u>SSLRandomSeed <i>context source [bytes]</i></u></a>	
Pseudo Random Number Generator (PRNG) seeding source	
<a href="#"><u>SSLRequire <i>expression</i></u></a>	
Allow access only when an arbitrarily complex boolean expression is true	
<a href="#"><u>SSLRequireSSL</u></a>	
Deny access when SSL is not used for the HTTP request	
<a href="#"><u>SSLSessionCache <i>type</i></u></a>	none
Type of the global/inter-process SSL Session Cache	

<a href="#">SSLSessionCacheTimeout <i>seconds</i></a>	300
Number of seconds before an SSL session expires in the Session Cache	
<a href="#">SSLUserName <i>varname</i></a>	
Variable name to determine user name	
<a href="#">SSLVerifyClient <i>level</i></a>	none
Type of Client Certificate verification	
<a href="#">SSLVerifyDepth <i>number</i></a>	1
Maximum depth of CA Certificates in Client Certificate verification	
<a href="#">StartServers <i>number</i></a>	
<a href="#">StartThreads <i>number</i></a>	
<a href="#">SuexecUserGroup <i>User Group</i></a>	
CGI	
<a href="#">ThreadLimit <i>number</i></a>	
<a href="#">ThreadsPerChild <i>number</i></a>	
<a href="#">ThreadStackSize <i>number</i></a>	65536
Determine the stack size for each thread	
<a href="#">TimeOut <i>seconds</i></a>	300
<a href="#">TraceEnable [<i>on off extended</i>]</a>	on
Determines the behaviour on TRACE requests	
<a href="#">TransferLog <i>file pipe</i></a>	
<a href="#">TypesConfig <i>file-path</i></a>	conf/mime.types
mime.types	
<a href="#">UnsetEnv <i>env-variable [env-variable] ...</i></a>	
<a href="#">UseCanonicalName <i>On Off Dns</i></a>	On
<a href="#">User <i>unix-userid</i></a>	#-1
ID	
<a href="#">UserDir <i>directory-filename</i></a>	public_html

<a href="#">VirtualDocumentRoot <i>interpolated-directory</i> none</a>	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">VirtualDocumentRootIP <i>interpolated-directory</i> none</a>	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">&lt;VirtualHost <i>addr[:port] [addr[:port]] ...&gt; ...</i></a> <a href="#">&lt;/VirtualHost&gt;</a>	
IP	
<a href="#">VirtualScriptAlias <i>interpolated-directory</i> none</a>	none
Dynamically configure the location of the CGI directory for a given virtual host	
<a href="#">VirtualScriptAliasIP <i>interpolated-directory</i> none</a>	none
Dynamically configure the location of the cgi directory for a given virtual host	
<a href="#">Win32DisableAcceptEx</a>	
accept() AcceptEx	
<a href="#">XBitHack on off full</a>	off
SSI	



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)



Apache

(MPMs)



---

## core

Apache HTTP

## mpm\_common

(MPM)

## beos

This Multi-Processing Module is optimized for BeOS.

## leader

An experimental variant of the standard worker MPM

## mpm\_netware

Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare

## mpmt\_os2

Hybrid multi-process, multi-threaded MPM for OS/2

## perchild

Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids

## prefork

fork

## threadpool

Yet another experimental variant of the standard worker MPM

## mpm\_winnt

Windows NT

## worker



A | C | D | E | F | H | I | L | M | N | P | R | S | U |  
V

### [mod\\_access](#)

IP

### [mod\\_actions](#)

CGI

### [mod\\_alias](#)

### [mod\\_asis](#)

HTTP

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

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

Allows "anonymous" user access to authenticated areas

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

Provides for user authentication using DBM files

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

User authentication using MD5 Digest Authentication.

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

Allows an LDAP directory to be used to store the database for HTTP Basic authentication.

### [mod\\_autoindex](#)

Unix ls Win32 dir

### [mod\\_cache](#)

Content cache keyed to URIs.

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

CERN httpd metafile semantics

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

CGI

### [mod\\_cgid](#)

CGI CGI

### [mod\\_charset\\_lite](#)

Specify character set translation or recoding

### [mod\\_dav](#)

([WebDAV](#))

### [mod\\_dav\\_fs](#)

[mod\\_dav](#)

### [mod\\_deflate](#)

### [mod\\_dir](#)

### [mod\\_disk\\_cache](#)

Content cache storage manager keyed to URIs

### [mod\\_dumpio](#)

Dumps all I/O to error log as desired.

### [mod\\_echo](#)

### [mod\\_env](#)

CGI SSI

### [mod\\_example](#)

Illustrates the Apache module API

### [mod\\_expires](#)

Expires Cache-Control HTTP

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

Pass the response body through an external program before delivery to the client

### [mod\\_file\\_cache](#)

Caches a static list of files in memory

### [mod\\_headers](#)

Customization of HTTP request and response headers

### [mod\\_imap](#)

Server-side imagemap processing

### [mod\\_include](#)

html (Server Side Includes)

### [mod\\_info](#)

### [mod\\_isapi](#)

ISAPI Extensions within Apache for Windows

### [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](#)

() (MIME )

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

Determines the MIME type of a file by looking at a few bytes of its contents

### [mod\\_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](#)

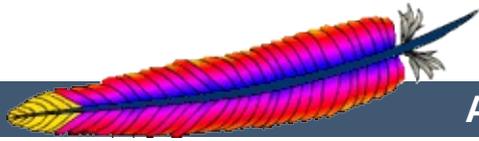
## [mod\\_usertrack](#)

*Clickstream* logging of user activity on a site

## [mod\\_version](#)

## [mod\\_vhost\\_alias](#)

Provides for dynamically configured mass virtual hosting



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



## Apache HTTP Server Version 2.0

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

## Frequently Asked Questions

The latest version of this FAQ is always available from the main Apache web site, at <<http://httpd.apache.org/docs/2.0/faq/>>. In addition, you can view this FAQ [all in one page](#) for easy searching and printing.

Since Apache 2.0 is quite new, we don't yet know what the *Frequently Asked Questions* will be. While this section fills up, you should also consult the [Apache 1.3 FAQ](#) to see if your question is answered there.



## [Support](#)

What do I do when I have problems?

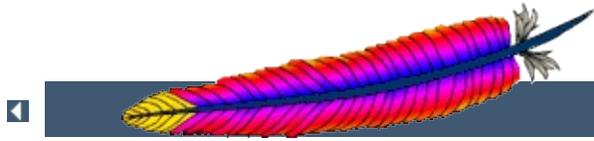
## [Error Messages](#)

What does this error message mean?

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

## Site Map

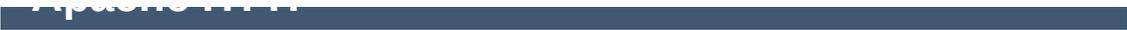
This translation may be out of date. Check the English version for recent changes.

[Apache HTTP 2.0](#)



- 
- [1.3 2.0](#)
  - [Apache 2.0](#)
  - [Apache License](#)





- [Apache](#)
- [Apache](#)
- 
- 
- [Directory, Location, Files](#)
- 
- 
- [URL](#)
- 
- [\(DSO\)](#)
- 
- 
- [Apache](#)
- [\(MPM\)](#)
- [Apache](#)
- [Apache](#)
- 
- [suEXEC](#)
- 
- [URL](#)



## Apache

- 
- 
- [IP](#)
- 
- [VirtualHost](#)
- 
- 
- [DNS Apache](#)





## Public SSL/TLS

- 
- [SSL/TLS :](#)
- [SSL/TLS :](#)
- [SSL/TLS :](#)
- [SSL/TLS : FAQ](#)



- 
- - 
  - [CGI](#)
  - [Server Side Includes](#)
  - [.htaccess](#)
  - 
  - [Apache](#)



- 
- - [Microsoft Windows Apache](#)
  - [Microsoft Windows Apache](#)
  - [Novell NetWare Apache](#)
  - [HPUX](#)
  - [EBCDIC Apache](#)



## Apache HTTPD

- 
- [: httpd](#)
- [: ab](#)
- [: apachectl](#)
- [: apxs](#)
- [: configure](#)
- [: dbmmanage](#)
- [: htdigest](#)
- [: htpasswd](#)
- [: logresolve](#)
- [: rotatelogs](#)
- [: suexec](#)
- 



## Apache

- 
- 
- [FIN\\_WAIT\\_2 Apache](#)
- 
- [Apache](#)
- 



- [Apache](#)
- [Apache](#)
  
- [Apache](#)
- [Apache MPM](#)
- [Apache MPM beos](#)
- [Apache MPM leader](#)
- [Apache MPM netware](#)
- [Apache MPM os2](#)
- [Apache MPM perchild](#)
- [Apache MPM prefork](#)
- [Apache MPM threadpool](#)
- [Apache MPM winnt](#)
- [Apache MPM worker](#)
  
- [Apache mod\\_access](#)
- [Apache mod\\_actions](#)
- [Apache mod\\_alias](#)
- [Apache mod\\_asis](#)
- [Apache mod\\_auth](#)
- [Apache mod\\_auth\\_anon](#)
- [Apache mod\\_auth\\_dbm](#)
- [Apache mod\\_auth\\_digest](#)
- [Apache mod\\_auth\\_ldap](#)
- [Apache mod\\_autoindex](#)
- [Apache mod\\_cache](#)
- [Apache mod\\_cern\\_meta](#)
- [Apache mod\\_cgi](#)
- [Apache mod\\_cgid](#)
- [Apache mod\\_charset\\_lite](#)
- [Apache mod\\_dav](#)
- [Apache mod\\_dav\\_fs](#)
- [Apache mod\\_deflate](#)

- [Apache mod\\_dir](#)
- [Apache mod\\_disk\\_cache](#)
- [Apache mod\\_dumpio](#)
- [Apache mod\\_echo](#)
- [Apache mod\\_env](#)
- [Apache mod\\_example](#)
- [Apache mod\\_expires](#)
- [Apache mod\\_ext\\_filter](#)
- [Apache mod\\_file\\_cache](#)
- [Apache mod\\_headers](#)
- [Apache mod\\_imap](#)
- [Apache mod\\_include](#)
- [Apache mod\\_info](#)
- [Apache mod\\_isapi](#)
- [Apache mod\\_ldap](#)
- [Apache mod\\_log\\_config](#)
- [Apache mod\\_log\\_forensic](#)
- [Apache mod\\_logio](#)
- [Apache mod\\_mem\\_cache](#)
- [Apache mod\\_mime](#)
- [Apache mod\\_mime\\_magic](#)
- [Apache mod\\_negotiation](#)
- [Apache mod\\_nw\\_ssl](#)
- [Apache mod\\_proxy](#)
- [Apache mod\\_proxy\\_connect](#)
- [Apache mod\\_proxy\\_ftp](#)
- [Apache mod\\_proxy\\_http](#)
- [Apache mod\\_rewrite](#)
- [Apache mod\\_setenvif](#)
- [Apache mod\\_so](#)
- [Apache mod\\_speling](#)
- [Apache mod\\_ssl](#)
- [Apache mod\\_status](#)
- [Apache mod\\_suexec](#)

- [Apache mod\\_unique\\_id](#)
- [Apache mod\\_userdir](#)
- [Apache mod\\_usertrack](#)
- [Apache mod\\_version](#)
- [Apache mod\\_vhost\\_alias](#)



---

- 

- [Apache API](#)

- [APR](#)

- [Apache 2.0](#)

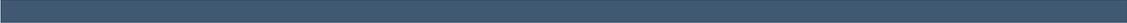
- [Apache 2.0](#)

- [Apache 1.3 Apache 2.0](#)

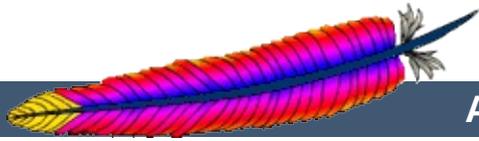
- [Apache 2.0](#)

- [Apache 2.0](#)





- 
- 
- 
- 



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



## Apache HTTP Server Version 2.0

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

## Server and Supporting Programs

This page documents all the executable programs included with the Apache HTTP Server.



## [httpd](#)

Apache hypertext transfer protocol server

## [apachectl](#)

Apache HTTP server control interface

## [ab](#)

Apache HTTP server benchmarking tool

## [apxs](#)

APache eXtenSion tool

## [configure](#)

Configure the source tree

## [dbmmanage](#)

Create and update user authentication files in DBM format for basic authentication

## [htdigest](#)

Create and update user authentication files for digest authentication

## [htdbm](#)

Manipulate DBM password databases.

## [htpasswd](#)

Create and update user authentication files for basic authentication

## [logresolve](#)

Resolve hostnames for IP-addresses in Apache logfiles

## [rotatelogs](#)

Rotate Apache logs without having to kill the server

## [suexec](#)

Switch User For Exec

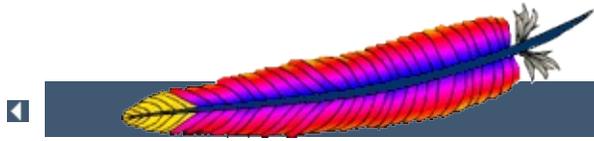
## [Other Programs](#)

Support tools with no own manual page.

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache SSL/TLS

Apache HTTP [mod\\_ssl](#) [OpenSSL](#) Secure Sockts Layer  
Transport Layer Security Ralf S. Engelschall mod\_ssl



## Documentation

- 
- 
- [How-To](#)
- 
- 

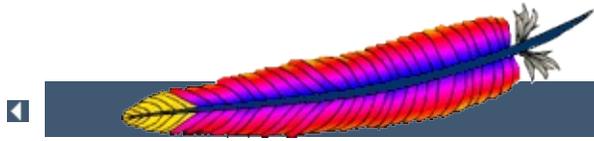


[mod\\_ssl](#)

---

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

| | [FAQ](#) | |



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

# Apache

This translation may be out of date. Check the English version for recent changes.

1 (www.company1.com and www.company2.c

IP [IP](#) IP

Apache IP

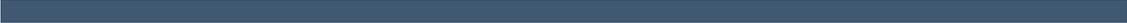
1.1 Apache

Apache 1.3

mod\_vhost\_alias

[IP](#)





- ( IP )
- IP ( IP )
- 
- (      )
- 
- 



- [<VirtualHost>](#)
- [NameVirtualHost](#)
- [ServerName](#)
- [ServerAlias](#)
- [ServerPath](#)

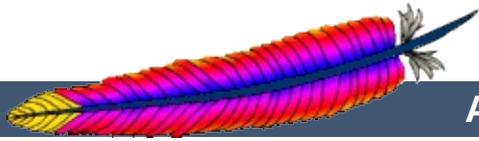
Apache

-S

```
/usr/local/apache2/bin/httpd -S
```

Apache

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](mailto:dev@httpd.apache.org) mailing list.



- [Apache 1.3 API Notes](#)
- [Apache 2.0 Hook Functions](#)
- [Request Processing in Apache 2.0](#)
- [How filters work in Apache 2.0](#)
- [Converting Modules from Apache 1.3 to Apache 2.0](#)
- [Debugging Memory Allocation in APR](#)
- [Documenting Apache 2.0](#)
- [Apache 2.0 Thread Safety Issues](#)

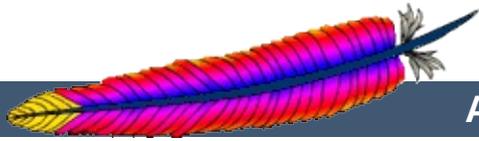


- 
- Tools provided by Ian Holsman:
    - [Apache 2 cross reference](#)
    - [Autogenerated Apache 2 code documentation](#)
  - Module Development Tutorials by Kevin O'Donnell
    - [Integrating a module into the Apache build system](#)
    - [Handling configuration directives](#)
  - [Some notes on 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.

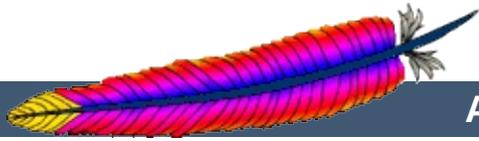
## [Relevant Standards](#)

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](#)



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



## Apache HTTP Server Version 2.0

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

## Platform Specific Notes



---

## Using Apache

This document explains how to install, configure and run Apache 2.0 under Microsoft Windows.

See: [Using Apache with Microsoft Windows](#)

## Compiling Apache

There are many important points before you begin compiling Apache. This document explain them.

See: [Compiling Apache for Microsoft Windows](#)



## Novell NetWare

This document explains how to install, configure and run Apache 2.0 under Novell NetWare 5.1 and above.

See: [Using Apache With Novell NetWare](#)

## EBCDIC

Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

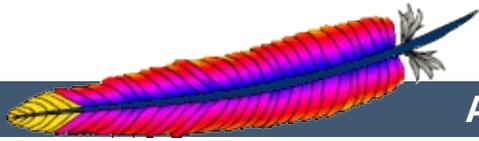
**Warning:** This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

See: [The Apache EBCDIC Port](#)

---

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](#) > [Programs](#)

## **suexec - Switch user before executing external programs**

suexec is used by the Apache HTTP Server to switch to another user before executing CGI programs. In order to achieve this, it must run as root. Since the HTTP daemon normally doesn't run as root, the suexec executable needs the setuid bit set and must be owned by root. It should never be writable for any other person than root.

For further information about the concepts and the security model of suexec please refer to the suexec documentation (<http://httpd.apache.org/docs/2.0/suexec.html>).



## suexec -V



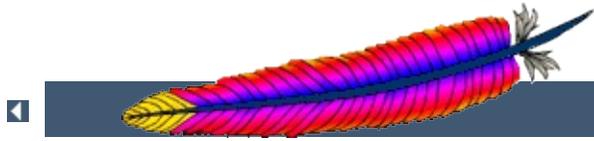
## **-V**

If you are root, this option displays the compile options of suexec. For security reasons all configuration options are changeable only at compile time.

---

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

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



| | [FAQ](#) | |



## Apache HTTP 2.0

[Apache](#) > [HTTP](#) > > [2.0](#)

## How-To /





: [\[redacted\]](#)

**CGI**

CGI (Common Gateway Interface)  
CGI

Apache

: [CGI:](#)

**.htaccess**

.htaccess

: [.htaccess](#)

**Server Side Includes**

SSI (Server Side Includes) HTML  
HTML

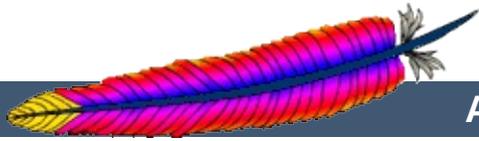
: [Server Side Includes \(SSI\)](#)

[UserDir](#)

http://example.com/~username/ " usern

[UserDir](#)

: [\(\\_public\\_html\)](#)



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



## Apache HTTP Server Version 2.0

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

## htdbm - Manipulate DBM password databases

htdbm is used to manipulate the DBM format files used to store usernames and password for basic authentication of HTTP users via [mod\\_auth\\_dbm](#). See the [dbmmanage](#) documentation for more information about these DBM files.

### See also

[httpd](#)

[dbmmanage](#)

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



**htdbm** [ **-TDBTYPE** ] [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-t** ] [ **-v** ] [ **-x** ] *filename username*

**htdbm -b** [ **-TDBTYPE** ] [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-t** ] [ **-v** ] *filename username password*

**htdbm -n** [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-t** ] [ **-v** ] *username*

**htdbm -nb** [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-t** ] [ **-v** ] *username password*

**htdbm -v** [ **-TDBTYPE** ] [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-t** ] [ **-v** ] *filename username*

**htdbm -vb** [ **-TDBTYPE** ] [ **-c** ] [ **-m** | **-d** | **-p** | **-s** ] [ **-t** ] [ **-v** ] *filename username password*

**htdbm -x** [ **-TDBTYPE** ] [ **-m** | **-d** | **-p** | **-s** ] *filename username*

**htdbm -l** [ **-TDBTYPE** ]



## **-b**

Use batch mode; *i.e.*, get the password from the command line rather than prompting for it. This option should be used with extreme care, since **the password is clearly visible** on the command line.

## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is rewritten and truncated. This option cannot be combined with the `-n` option.

## **-n**

Display the results on standard output rather than updating a database. This option changes the syntax of the command line, since the *passwdfile* argument (usually the first one) is omitted. It cannot be combined with the `-c` option.

## **-m**

Use MD5 encryption for passwords. On Windows, Netware and TPF, this is the default.

## **-d**

Use `crypt ( )` encryption for passwords. The default on all platforms but Windows, Netware and TPF. Though possibly supported by `htdbm` on all platforms, it is not supported by the [httpd](#) server on Windows, Netware and TPF.

## **-s**

Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (Idif).

## **-p**

Use plaintext passwords. Though `htdbm` will support creation on all platforms, the [httpd](#) daemon will only accept plain text passwords on Windows, Netware and TPF.

**-l**

Print each of the usernames and comments from the database on stdout.

**-t**

Interpret the final parameter as a comment. When this option is specified, an additional string can be appended to the command line; this string will be stored in the "Comment" field of the database, associated with the specified username.

**-v**

Verify the username and password. The program will print a message indicating whether the supplied password is valid. If the password is invalid, the program exits with error code 3.

**-x**

Delete user. If the username exists in the specified DBM file, it will be deleted.

***filename***

The filename of the DBM format file. Usually without the extension `.db`, `.pag`, or `.dir`. If `-c` is given, the DBM file is created if it does not already exist, or updated if it does exist.

***username***

The username to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.

***password***

The plaintext password to be encrypted and stored in the DBM file. Used only with the `-b` flag.

***-TDBTYPE***

Type of DBM file (SDBM, GDBM, DB, or "default").



One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, GNU GDBM, and Berkeley/Sleepycat DB 2/3/4. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by *filename* is the same format that `htdbm` expects to see. `htdbm` currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.



htdbm returns a zero status ("true") if the username and password have been successfully added or updated in the DBM File. htdbm returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn't match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the [Restrictions section](#)), and 7 if the file is not a valid DBM password file.



## Examples

```
htdbm /usr/local/etc/apache/.htdbm-users jsmith
```

Adds or modifies the password for user `jsmith`. The user is prompted for the password. If executed on a Windows system, the password will be encrypted using the modified Apache MD5 algorithm; otherwise, the system's `crypt ( )` routine will be used. If the file does not exist, `htdbm` will do nothing except return an error.

```
htdbm -c /home/does/public_html/.htdbm jane
```

Creates a new file and stores a record in it for user `jane`. The user is prompted for the password. If the file exists and cannot be read, or cannot be written, it is not altered and `htdbm` will display a message and return an error status.

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

Encrypts the password from the command line (`Pwd4Steve`) using the MD5 algorithm, and stores it in the specified file.



## Security Considerations

Web password files such as those managed by `htdbm` should *not* be within the Web server's URI space -- that is, they should not be fetchable with a browser.

The use of the `-b` option is discouraged, since when it is used the unencrypted password appears on the command line.



On the Windows and MPE platforms, passwords encrypted with `htdbm` are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

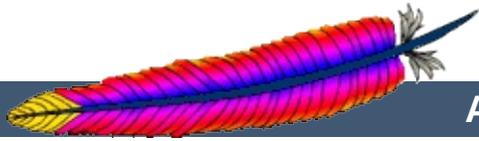
The MD5 algorithm used by `htdbm` is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Username are limited to 255 bytes and may not include the character `:`.

---

Copyright 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

*``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](#)

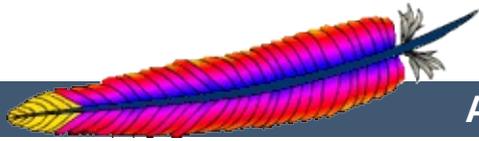


Extensive documentation on the directives provided by this module is provided in the [mod\\_rewrite reference documentation](#).

---

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.



## Time-Dependent Redirecting

### 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 $_;
}
```

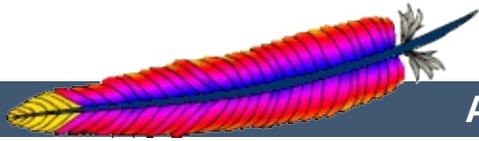
This is a demonstration-only example and just rewrites all URLs `/~quux/foo/...` to `/~quux/bar/...`. Actually you can program whatever you like. But notice that while such

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 `cgiwrap` 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 `cgiwrap` needs URLs in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uqe]/([^\s/]+)/\.\www/(.+)\.scgi(.*)\. . .  
... /internal/cgi/user/cgiwrap/~$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, $mtime) = stat($fil
    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
##
```

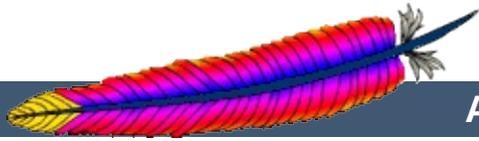
```
http://www.badguys.com/bad/index.html -  
http://www.badguys.com/bad/index2.html -  
http://www.badguys.com/bad/index3.html http://somewhere.co
```

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.

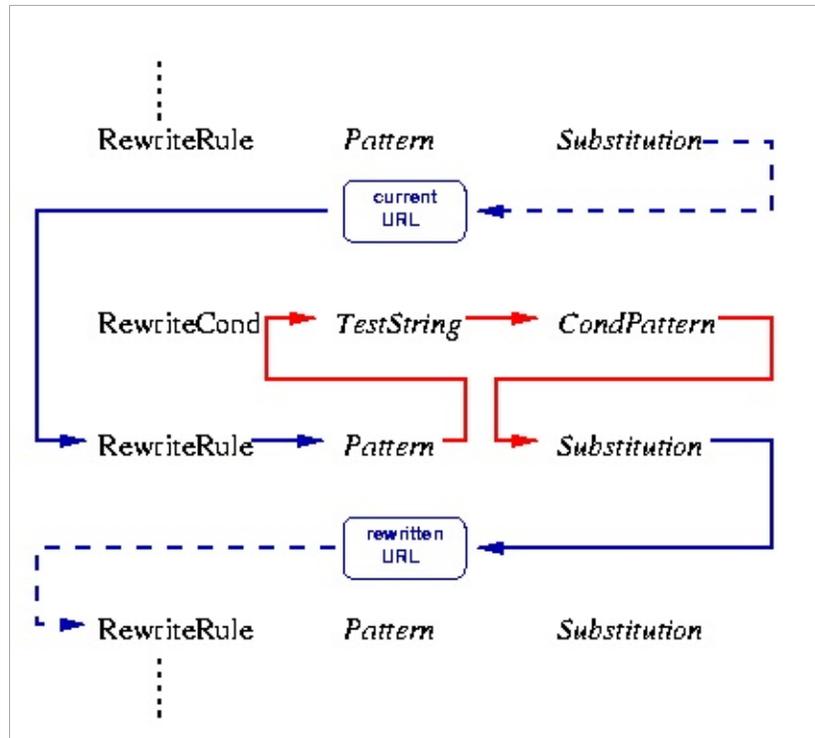
Don't forget these two points!



## 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.



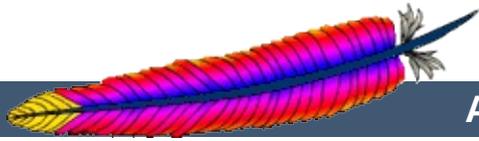
**Figure**

**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](#)



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



## Apache HTTP Server Version 2.0

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

## Frequently Asked Questions

The latest version of this FAQ is always available from the main Apache web site, at <<http://httpd.apache.org/docs/2.0/faq/>>.

Since Apache 2.0 is quite new, we don't yet know what the *Frequently Asked Questions* will be. While this section fills up, you should also consult the [Apache 1.3 FAQ](#) to see if your question is answered there.



---

## Support

What do I do when I have problems?

## Error Messages

What does this error message mean?



- ["Why can't I ...? Why won't ... work?" What to do in case of problems](#)
- [Whom do I contact for support?](#)

## "Why can't I ...? Why won't ... work?" What to do in case of problems

If you are having trouble with your Apache server software, you should take the following steps:

### Check the errorlog!

Apache tries to be helpful when it encounters a problem. In many cases, it will provide some details by writing one or messages to the server error log. Sometimes this is enough for you to diagnose & fix the problem yourself (such as file permissions or the like). The default location of the error log is `/usr/local/apache2/logs/error_log`, but see the [ErrorLog](#) directive in your config files for the location on your server.

### Check the [FAQ](#)!

The latest version of the Apache Frequently-Asked Questions list can always be found at the main Apache web site.

### Check the Apache bug database

Most problems that get reported to The Apache Group are recorded in the [bug database](#). **Please** check the existing reports, open **and** closed, before adding one. If you find that your issue has already been reported, please *don't* add a "me, too" report. If the original report isn't closed yet, we suggest that you check it periodically. You might also consider contacting the original submitter, because there may be an email exchange going on about the issue that isn't getting recorded in the database.

## Ask in a user support forum

Apache has an active community of users who are willing to share their knowledge. Participating in this community is usually the best and fastest way to get answers to your questions and problems.

[Users mailing list](#)

[#httpd](#) on [Freenode IRC](#) is available for user support issues.

USENET newsgroups:

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

## If all else fails, report the problem in the bug database

If you've gone through those steps above that are appropriate and have obtained no relief, then please *do* let the httpd developers know about the problem by [logging a bug report](#).

If your problem involves the server crashing and generating a core dump, please include a backtrace (if possible). As an example,

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

(Substitute the appropriate locations for your *ServerRoot* and your *httpd* and *core* files. You may have to use *gdb* instead of *dbx*.)

## Whom do I contact for support?

With several million users and fewer than forty volunteer developers, we cannot provide personal support for Apache. For free support, we suggest participating in a [user forum](#).



- [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**

Apache uses the `sendfile` syscall on platforms where it is available in order to speed sending of responses. Unfortunately, on some systems, Apache will detect the presence of `sendfile` at compile-time, even when it does not work properly. This happens most frequently when using network or other non-standard file-system.

Symptoms of this problem include the above message in the error log and zero-length responses to non-zero-sized files. The problem generally occurs only for static files, since dynamic content usually does not make use of `sendfile`.

To fix this problem, simply use the [EnableSendfile](#) directive to disable `sendfile` for all or part of your server. Also see the [EnableMMAP](#), which can help with similar problems.

## **AcceptEx Failed**

If you get error messages related to the `AcceptEx` syscall on win32, see the [Win32DisableAcceptEx](#) directive.

## **Premature end of script headers**

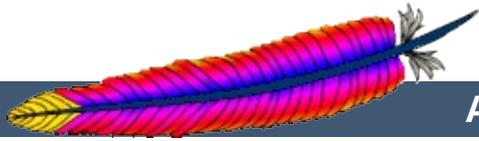
Most problems with CGI scripts result in this message written in the error log together with an `Internal Server Error` delivered to the browser. A guide to helping debug this type of

problem is available in the [CGI tutorial](#).

---

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 Thread Safety Issues

When using any of the threaded mpms in Apache 2.0 it is important that every function called from Apache be thread safe. When linking in 3rd party extensions it can be difficult to determine whether the resulting server will be thread safe. Casual testing generally won't tell you this either as thread safety problems can lead to subtle race conditons that may only show up in certain conditions under heavy load.



## Global and Static Variables

When writing your module or when trying to determine if a module or 3rd party library is thread safe there are some common things to keep in mind.

First, you need to recognize that in a threaded model each individual thread has its own program counter, stack and registers. Local variables live on the stack, so those are fine. You need to watch out for any static or global variables. This doesn't mean that you are absolutely not allowed to use static or global variables. There are times when you actually want something to affect all threads, but generally you need to avoid using them if you want your code to be thread safe.

In the case where you have a global variable that needs to be global and accessed by all threads, be very careful when you update it. If, for example, it is an incrementing counter, you need to atomically increment it to avoid race conditions with other threads. You do this using a mutex (mutual exclusion). Lock the mutex, read the current value, increment it and write it back and then unlock the mutex. Any other thread that wants to modify the value has to first check the mutex and block until it is cleared.

If you are using [APR](#), have a look at the `apr_atomic_*` functions and the `apr_thread_mutex_*` functions.



This is a common global variable that holds the error number of the last error that occurred. If one thread calls a low-level function that sets `errno` and then another thread checks it, we are bleeding error numbers from one thread into another. To solve this, make sure your module or library defines `_REENTRANT` or is compiled with `-D_REENTRANT`. This will make `errno` a per-thread variable and should hopefully be transparent to the code. It does this by doing something like this:

```
#define errno (*(__errno_location()))
```

which means that accessing `errno` will call `__errno_location()` which is provided by the `libc`. Setting `_REENTRANT` also forces redefinition of some other functions to their `*_r` equivalents and sometimes changes the common `getc/putc` macros into safer function calls. Check your `libc` documentation for specifics. Instead of, or in addition to `_REENTRANT` the symbols that may affect this are `_POSIX_C_SOURCE`, `_THREAD_SAFE`, `_SVID_SOURCE`, and `_BSD_SOURCE`.



---

Not only do things have to be thread safe, but they also have to be reentrant. `strtok()` is an obvious one. You call it the first time with your delimiter which it then remembers and on each subsequent call it returns the next token. Obviously if multiple threads are calling it you will have a problem. Most systems have a reentrant version of the function called `strtok_r()` where you pass in an extra argument which contains an allocated `char *` which the function will use instead of its own static storage for maintaining the tokenizing state. If you are using [APR](#) you can use `apr_strtok()`.

`crypt()` is another function that tends to not be reentrant, so if you run across calls to that function in a library, watch out. On some systems it is reentrant though, so it is not always a problem. If your system has `crypt_r()` chances are you should be using that, or if possible simply avoid the whole mess by using `md5` instead.



The following is a list of common libraries that are used by 3rd party Apache modules. You can check to see if your module is using a potentially unsafe library by using tools such as `ldd(1)` and `nm(1)`. For [PHP](#), for example, try this:

```
% ldd libphp4.so
libsablot.so.0 => /usr/local/lib/libsablot.so.0 (0x401f6000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x402da000)
libsnmp.so.0 => /usr/lib/libsnmp.so.0 (0x402f9000)
libpdf.so.1 => /usr/local/lib/libpdf.so.1 (0x40353000)
libz.so.1 => /usr/lib/libz.so.1 (0x403e2000)
libpng.so.2 => /usr/lib/libpng.so.2 (0x403f0000)
libmysqlclient.so.11 => /usr/lib/libmysqlclient.so.11
(0x40411000)
libming.so => /usr/lib/libming.so (0x40449000)
libm.so.6 => /lib/libm.so.6 (0x40487000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x404a8000)
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x404e7000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40505000)
libssl.so.2 => /lib/libssl.so.2 (0x40532000)
libcrypto.so.2 => /lib/libcrypto.so.2 (0x40560000)
libresolv.so.2 => /lib/libresolv.so.2 (0x40624000)
libdl.so.2 => /lib/libdl.so.2 (0x40634000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40637000)
libc.so.6 => /lib/libc.so.6 (0x4064b000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
```

In addition to these libraries you will need to have a look at any libraries linked statically into the module. You can use `nm(1)` to look for individual symbols in the module.



Please drop a note to [dev@httpd.apache.org](mailto:dev@httpd.apache.org) if you have additions or corrections to this list.

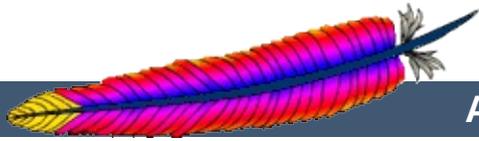
Library	Version	Thread Safe?	Notes
<a href="#">ASpell/PSpell</a>		?	
<a href="#">Berkeley DB</a>	3.x, 4.x	Yes	Be careful about sharing a connection across threads.
<a href="#">bzip2</a>		Yes	Both low-level and high-level APIs are thread-safe. However, high-level API requires calling <code>errno</code> .
<a href="#">cdb</a>		?	
<a href="#">C-Client</a>		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.
<a href="#">cpdflib</a>		?	
<a href="#">libcrypt</a>		?	
<a href="#">Expat</a>		Yes	Need a separate parser instance per thread.
<a href="#">FreeTDS</a>		?	
<a href="#">FreeType</a>		?	
<a href="#">GD 1.8.x</a>		?	
<a href="#">GD 2.0.x</a>		?	
<a href="#">gdbm</a>		No	Errors returned via a static <code>gdbm</code> pointer.
<a href="#">ImageMagick</a>	5.2.2	Yes	ImageMagick docs claim it is thread-safe starting in version 5.2.2 (see <a href="#">Change log</a> ).
<a href="#">Imlib2</a>		?	
<a href="#">libjpeg</a>	v6b	?	

<a href="#">libmysqlclient</a>		Yes	Use mysqlclient_r library variant for thread safety. For more information, please see <a href="http://www.mysql.com/doc/en/Thread-Safety.html">http://www.mysql.com/doc/en/Thread-Safety.html</a>
<a href="#">Ming</a>	0.2a	?	
<a href="#">Net-SNMP</a>	5.0.x	?	
<a href="#">OpenLDAP</a>	2.1.x	Yes	Use ldap_r library variant to ensure thread safety
<a href="#">OpenSSL</a>	0.9.6g	Yes	Requires proper usage of CRYPTO_THREADID. See <a href="http://www.openssl.org/docs/crypto/crypto_threadid.html">http://www.openssl.org/docs/crypto/crypto_threadid.html</a> for details. CRYPTO_set_locking_callback and CRYPTO_set_id_callback are deprecated.
<a href="#">liboci8 (Oracle 8+)</a>	8.x,9.x	?	
<a href="#">pdflib</a>	5.0.x	Yes	PDFLib docs claim it is thread safe. However, the docs indicate it has been partially threaded since version V1.91: <a href="http://www.pdflib.com/products/pdflib.html">http://www.pdflib.com/products/pdflib.html</a>
<a href="#">libpng</a>	1.0.x	?	
<a href="#">libpng</a>	1.2.x	?	
<a href="#">libpq (PostgreSQL)</a>	7.x	Yes	Don't share connections across threads. Avoid out for crypt ( ) calls
<a href="#">Sablotron</a>	0.95	?	
<a href="#">zlib</a>	1.1.4	Yes	Relies upon thread-safe zalloc and zfree. Default is to use libc's calloc/free and free. For thread safety, use zlib's zcalloc and zcfree.

---

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](#)