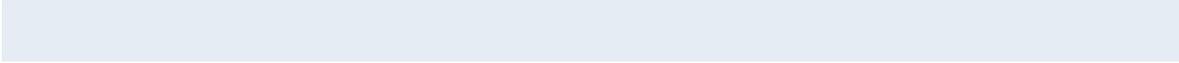


Apache HTTP Server 2.2

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

| | | 200614 |

Apache HTTP Server Version 2.2 [2006321]



[Apache 2.1/2.2](#)

[Apache 2.0](#)

[2.0 2.2](#)

[Apache](#)



(MPM)



[\(DSO\)](#)

[URL](#)

[SSL/TLS](#)

[CGISuexec](#)

[URL](#)

.../

[...](#)

[CGI](#)

[.htaccess](#)

[\(SSI\)](#)

[\(public_html\)](#)

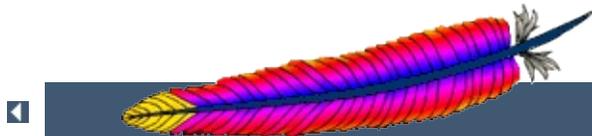


[Microsoft Windows](#)

[Novell NetWare](#)

[EBCDIC Port](#)





Apache HTTP Server 2.2

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

| | | 200617 |

1.32.0

Apache

src/CHANGES



-
- Apache autoconf libtool Apache 1.3 APACI
 - Apache 2.0 (MPM)



- Apache1.3MPMApache1.3 [preforkMPM](#)
- [proxy module](#)HTTP/1.1 [<Proxy>](#)<Directory proxy:>
- PATH_INFO () PATH_INFO [INCLUDES](#)PHP
PATH_INFO [AcceptPathInfo](#)PATH_INFO PATH_INFO
- [CacheNegotiatedDocs](#) On Off
[CacheNegotiatedDocs](#)CacheNegotiatedDocs on
- [ErrorDocument](#)

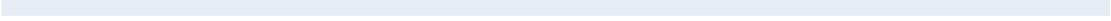
```
ErrorDocument 403 "Some Message"
```

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

URL

- AccessConfig ResourceConfig [Include](#)
" Include conf/access.conf" Include
conf/srm.conf" httpd.confApache [Include](#)
httpd.conf srm.confaccess.conf
- BindAddressPort [Listen](#)
- Apache1.3PortURLApache2.0 [ServerName](#)URL
- ServerTypeMPMinetd()MPM
- mod_log_agentmod_log_referer [CustomLog](#)
[mod_log_config](#)
- AddModuleClearModuleListApache2.0 API
- FancyIndexing [IndexOptions](#)FancyIndexing
- [mod_negotiation](#)MultiViews [MultiviewsMatch](#)
- (2.0.51) ErrorHeader[Header](#)

```
Header always set foo bar
```

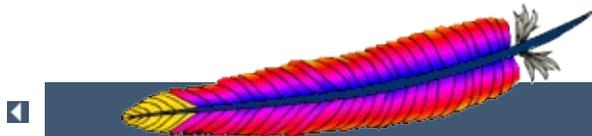


-
- [Apache1.3mod_auth_digest](#)
 - [Apache1.3mod_mmap_staticmod_file_cache](#)
 - src



Apache2.0APIApache1.3 Apache2.0

||||



Apache HTTP Server 2.2

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

| | | 200617 |

2.02.2

Apache

src/CHANGES

2.0 2.2 1.3

[1.32.0](#)



2.0 configure(build/config.nice)

- mod_imap [mod_imagemap](#)
- mod_auth
[mod_auth_basic](#)[mod_authn_file](#)[mod_authz_user](#)[mod_authz](#)
- mod_access [mod_authz_host](#)
- mod_auth_ldap [mod_authnz_ldap](#)
- APR 1.0 API
- PCRE5.0



2.02.2 LoadModule

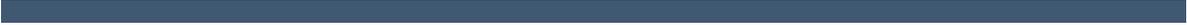
2.2 conf/extra/ conf/original

- `apachectl startssl` `SSL` `httpd.conf` `mod_ssl`
`apachectl start` `mod_ssl` `conf/extra/httpd-ssl.conf`
- `UseCanonicalName` `Off` `UseCanonicalName` `On`
- `UserDir` `mod_userdir` `UserDir`
`public_html`



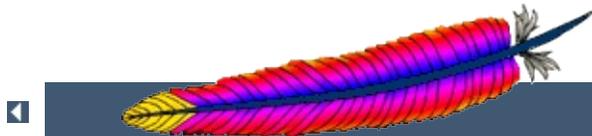
-
- [mod_cache2.0](#)
 - [mod_disk_cache2.0](#)
 - [mod_mem_cache2.0](#)
 - [mod_charset_lite2.0](#)
 - [mod_dumpio2.0](#)





2.02.2

|||



Apache HTTP Server 2.2

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

| | | 200615 |

Apache 2.2

Apache HTTP Server 2.0.2.2 1.3

[Apache 2.0](#)



!(Authn/Authz)

(authentication)(authorization) [mod_authn_alias](#)

[mod_cache](#)[mod_disk_cache](#)[mod_mem_cache](#)

[htcacheclean](#)[mod_disk_cache](#)

Apache

(Graceful stop)

[prefork](#)[worker](#)[event](#)(MPM)[httpd](#)[graceful-stop](#)

[GracefulShutdownTimeout](#) [httpd](#)

[mod_proxy_balancer](#)[mod_proxy](#) [mod_proxy_ajp](#)[Apache](#)

[Tomcat](#)Apache JServ Protocol version 1.3

5.0[Perl](#)(PCRE) [httpd](#) --with-pcre PCRE

[mod_filter](#)Apache2.0

httpd32Unix2GB2G(request body)

Event MPM

[event](#)(MPM)(Keep Alive)httpd(worker)(/)

SQL

[mod_dbd](#)apr_dbd(framework)MPM

WindowswindowsApacheWindows



/(Authn/Authz)

aaa(digest authentication)mod_auth [mod_auth_basic](#)
[mod_authn_file](#)mod_auth_dbm [mod_authn_dbm](#)
mod_access [mod_authz_host](#)[mod_authn_alias](#)

[mod_authnz_ldap](#)

2.0mod_auth_ldap2.2Authn/AuthzLDAP [Require](#)

[mod_info](#)

?config Apache(request hook) httpd -V

[mod_ssl](#)

[RFC 2817](#)TLS

[mod_imagemap](#)

mod_imap[mod_imagemap](#)



[httpd](#)

-M -l mod_soDSO()

[httxt2dbm](#)

dbm RewriteMapdbm(map)



APR 1.0 API

Apache2.2 APR 1.0 API APR APR-Util APR

/(Authn/Authz)

- mod_auth_* -> HTTP
- mod_authn_* ->
- mod_authz_* -> ()
- mod_authnz_* ->

ap_log_cerrorIP

(hook)

test_config httpd -t

MPM

ThreadStackSizeMPM

ap_register_output_filter_protocol
ap_filter_protocolmod filter

(Monitor hook)

API

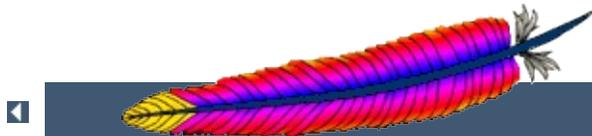
pcreposix.h ap_regex.h POSIX.2 regex.h ap_
(ap_regex.h) regcomp, regexcap_regcomp,
ap_regcomp

DBD(SQLAPI)

1.x2.0SQLApache 2.1 ap_dbd API (MPM)
APR 1.2 apr_dbd API

API API

||||



Apache HTTP Server 2.2

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

| | | 2006321 |

Apache 2.0

Apache 1.32.0



Unix

POSIXUnixApache()

autoconflibtoolApache

Apache [mod_echo](#)

Unix

Apache2.0BeOSOS/2WindowsUnix [\(MPM\)](#)Apache(APR)

ApacheAPIPOSIXbug

Apache API

2.0API1.32.0per-hookApache

IPv6

Apache(APR library)IPv6ApacheIPv6

```
ListenNameVirtualHostVirtualHostIPv6\(  
" Listen [2001:db8::1]:8080")
```

Apache [mod_include](#)INCLUDESCGI
[mod_ext_filter](#)CGI

SSI

PortBindAddressIP [Listen](#) [ServerName](#)

Windows NT Unicode

Apache2.0WindowsNTutf-8UnicodeWindowsNT(
Windows2000/XP/2003)

Windows95/98/ME

Apache2.0[Perl\(PCRE\)](#)Perl 5



mod_ssl

Apache2.0OpenSSLSSL/TLS

mod_dav

Apache2.0HTTPweb

mod_deflate

Apache2.0

mod_auth_ldap

Apache2.0.41LDAPHTTP [mod_ldap](#)

mod_auth_digest

mod_charset_lite

Apache2.0

mod_file_cache

Apache2.0Apache1.3 [mod_mmap_static](#)

mod_headers

Apache2.0 [mod_proxy](#)

mod_proxy

HTTP/1.1 [<Proxy>\(\)](#) [<Directory](#)
"proxy:..."> [proxy_connect](#)[proxy_ftpproxy_http](#)

mod_negotiation

[ForceLanguagePriority](#) [MultiViews](#)

mod_autoindex

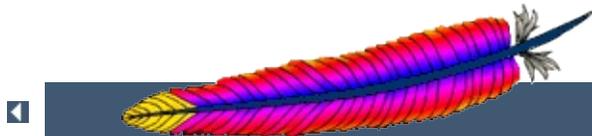
HTML

mod_include

SSI(Perl) [mod_include](#) \$0 .. \$9

mod_auth_dbm

[AuthDBMTypeDBM](#)



||||



Apache HTTP Server 2.2

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

|| < > | ??? |

The Apache License, Version 2.0

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

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions

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

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

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

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

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

"Object" form shall mean any form resulting from mechanical

transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

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

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

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

"Contributor" shall mean Licensor and any individual or Legal

Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

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

notices stating that You changed the files; and

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

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

5. Submission of Contributions. Unless You explicitly state

otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

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

losses), even if such Contributor has been advised of the possibility of such damages.

- 9. Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

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

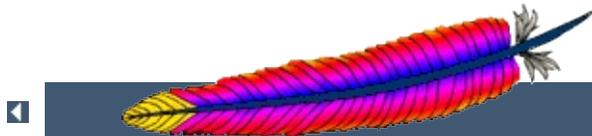
To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend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

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

```
Licensed under the Apache License, Version 2.0 (the  
you may not use this file except in compliance with  
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 implied. See the License for the specific language governing permissions and limitations under the License.



Apache HTTP Server 2.2

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

| | | 200614 |



ApacheUnixUnixWindows [Microsoft WindowsApache](#)

Apache libtoolautoconf

(2.2.54 → 2.2.55)



\$ lynx http://httpd.apache.org/download.cgi
\$ gzip -d httpd- <i>NN</i> .tar.gz
\$ tar xvf httpd- <i>NN</i> .tar
\$ cd httpd- <i>NN</i>
\$./configure --prefix= <i>PREFIX</i>
\$ make
\$ make install
\$ vi <i>PREFIX</i> /conf/httpd.conf
\$ <i>PREFIX</i> /bin/apachectl -k start

NN *PREFIX* *PREFIX*/usr/local/apache2

Apache httpd



Apache

50MBApache10MB

ANSI-C

ANSI-C [\(FSF\)GCCGCCANSI](#) PATHmake

HTTP(NTP) ntpdatextntpdNTP [NTP](#)

Perl 5 []

Perl [apxsdbmmanage](#) Perl5(5.003)PerlPerl 4Perl 5
--with-perl [configure](#) [configure](#)Perl 5Apache
httpd

[apr/apr-util >= 1.2](#)

aprapr-utilApache httpd aprapr-util1.01.1
apr/apr-util1.2httpd apr/apr-util

```
# apr 1.2
cd srclib/apr
./configure --prefix=/usr/local/apr-httpd/
make
make install

# apr-util 1.2
cd ../apr-util
./configure --prefix=/usr/local/apr-util-
httpd/ --with-apr=/usr/local/apr-httpd/
make
make install

# httpd
cd ../../
./configure --with-apr=/usr/local/apr-httpd/ -
-with-apr-util=/usr/local/apr-util-httpd/
```



Apache [Apache HTTP](#)UNIXApache()
INSTALL.bindist

tar [PGP](#)([PGP](#))



Apache httpd tar

```
$ gzip -d httpd-NN.tar.gz  
$ tar xvf httpd-NN.tar
```

cd



```
configure(Apache CVS      autoconflibtool buildconf  
)
```

```
./configure configure
```

```
Apache --prefix Apache
```

```
ApacheBaseApache --enable-module module  
" mod_" --enable-module=shared (DSO) --  
disable-module Base configure
```

```
configure configure
```

```
Apache /sw/pkg/apache mod\_rewrite mod\_speling  
DSO
```

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

```
configureMakefile
```



Apache

\$ make



PREFIX(--prefix)

```
$ make install
```



PREFIX/conf/Apache HTTP

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

[docs/manual/Apache](#) <http://httpd.apache.org/docs/2.2/>



Apache HTTP

```
$ PREFIX/bin/apachectl -k start
```

http://localhost/ DocumentRoot PREFIX/htdocs/

```
$ PREFIX/bin/apachectl -k stop
```



(release announcement)CHANGES(1.3 → 2.02.0 → 2.2)API

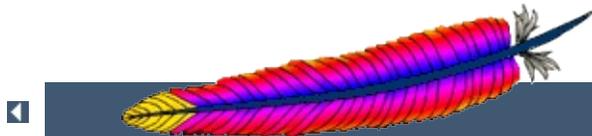
(2.2.55 → 2.2.57) make install [configure](#)
API [configure](#)

buildconfig.nice [configure](#) config.nice

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

Apache	--prefix	Listen
--------	----------	------------------------

||||



Apache HTTP Server 2.2

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

| | | 200614 |

Apache

Windows NT/2000/XP/2003ApacheWindows 95/98/ME

[Apache](#)[Apache](#)

Unix [httpd](#) [httpd](#)



Listen80(1024)Apacheroot httpdroot

httpdapachectl httpd httpd apachectl httpd
apachectl apachectl HTTPDhttpd

httpdhttpd.conf -f

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

DocumentRoot



Apache ErrorLog" Unable to bind to Port ..."

- root
- Apacheweb



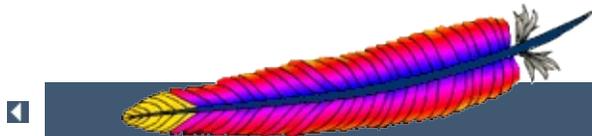
[apachectl](#)(rc.localrc.N)rootApache

[apachectl](#)SysV startrestartstop [httpd](#) [apachectl](#)



[httpd.apachectl](#)Apache





Apache HTTP Server 2.2

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

| | | 200616 |



UnixApache Windows NT/2000/XP/2003 [Apache](#) Windows
9x/ME [Apache](#)



Apache [httpd](#)UNIX kill [httpd](#) [PidFile](#)PID
[TERM](#)HUPUSR:

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

```
httpd -k    stoprestartgracefulgraceful-stop  
apachectlhttpd
```

```
httpd
```

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

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



TERM

`apachectl -k stop`

TERMstop



USR1

apachectl -k graceful

USR1graceful()

MPM StartServers StartServers
StartServers

mod_statusUSR1 () *scoreboard*

mod_status" G"

USR1 USR11015

Apache("") -t (
[httpd](#))root [httpd](#)root([httpd](#))



HUP

apachectl -k restart

HUPrestartTERM

mod_statusHUP



WINCH

apachectl -k graceful-stop

WINCHgraceful-stop() [PidFile](#)
[GracefulShutdownTimeout](#) TERM

"" TERM [PidFile](#) apachectlhttpd

```
graceful-stophttpdApache  
LockfileScriptSockPID CGI httpd  
rotatelogs rotatelogs
```



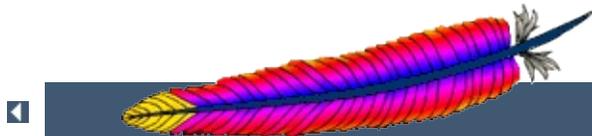
Apache 1.2b9

""

ScoreBoardFileScoreBoard"bind: Address already in use"
(HUP)"long lost child came home!"(USR1)ScoreBoard
ScoreBoard

HTTP(KeepAlive)1.220

| | | |



Apache HTTP Server 2.2

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

| | | 200611 |



Apache



<code>mod_mime</code>	<code><IfDefine></code> <code> <u>Include</u></code> <code> <u>TypesConfig</u></code>

Apache httpd.conf -f IncludeApache

MIME TypesConfig mime.types



Apache"\"()

(argument)"#"

apachectl configtest -t Apache



<code>mod_so</code>	<code><IfModule></code> <code>LoadModule</code>

Apache [base](#) [DSO](#) [LoadModule](#)Apache
`<IfModule>`

-1



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

<Directory><DirectoryMatch><Files><FilesMatch><LocationMatch>
URL

Apache <VirtualHost>()





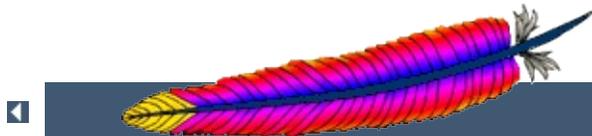
	<u>AccessFileName</u>
	<u>AllowOverride</u>

Apache .htaccess AccessFileName .htaccess
.htaccess .htaccess

.htaccess AllowOverride.htaccess

.htaccess .htaccess

||||



Apache HTTP Server 2.2

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

| | | 200615 |

()

URL()

.htaccess



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

<IfDefine><IfModule><IfVersion>

<IfDefine>[httpd](#) [httpd -DClosedForNow](#)

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

<IfModule>() LoadModule

MimeMagicFiles[mod_mime_magic](#)

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

<IfVersion><IfDefine><IfModule>[httpd](#)

```
<IfVersion >= 2.1>  
  # 2.1.0  
</IfVersion>
```

<IfDefine><IfModule><IfVersion>!"



```
UnixApache          /usr/local/apache2 Windows
Apache              "C:/Program Files/Apache Group/Apache2"(
ApacheWindows)web
/usr/local/apache2/htdocs/dir/
```

```
<Directory><Files>(<DirectoryMatch><FilesMatch>)
  <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>( <LocationMatch>)" /private"URL
    http://yoursite.example.com/privatehttp://yoursite.e
" /private"URL
```

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

```
<Location>URLApache mod_status server-status
```

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

```
<Directory><Files><Location>Cfnmatchshell"*""?"
"[ seq]" seq"/"
```

```
    <DirectoryMatch><FilesMatch><LocationMatch>
Perl
```

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

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

```
Deny from all
</FilesMatch>
```

<Directory><Files> <Location>

<Location>

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

```
http://yoursite.example.com/dir/
http://yoursite.example.com/DIR/
Unix()
```

<Directory>

```
<Location />URLURL
```



<VirtualHost>



`<Proxy><ProxyMatch>mod_proxyURL` `cnn.com`

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



<Directory>
<DirectoryMatch><Files><FilesMatch><Location><Location>

- AllowOverride<Directory>
- OptionsFollowSymLinksSymLinksIfOwnerMatch
<Directory> .htaccess
- Options<Files><FilesMatch>



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

```
<Directory> <Directory>(1) <Directory  
/var/web/dir><Directory /var/web/dir/subdir>  
<Directory> IncludeInclude
```

```
<VirtualHost>
```

```
mod_proxy <Proxy><Directory>
```

```
( AliasesDocumentRootsURL)  
<Location>/<LocationMatch>
```

```
A > B > C > D > E
```

```
<Location />  
E  
</Location>  
  
<Files f.html>
```

```
D
</Files>

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

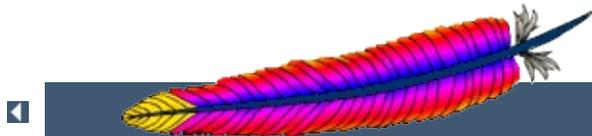
<DirectoryMatch "^.*b$">
C
</DirectoryMatch>

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

<Directory> <Location>

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

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



Apache HTTP Server 2.2

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

| | | 200611 |



m
od cachemod disk cachemod mem cachemod file cache
htcachecleanApacheweb(proxy)



Apache2.2 mod_cachemod_file_cacheweb(origin
webserver)(proxy)HTTP

mod_cachemod_mem_cachemod_disk_cacheHTTP(content)
mod_cacheHTTP mod_cache

mod_file_cacheURL mod_file_cache(file-handle)
(memory-mapping)Apache

mod_file_cache CacheFileMMapStatic mod_cache

HTTP URL



mod_cache	CacheEnable
mod_mem_cache	CacheDisable
mod_disk_cache	MMapStatic
mod_file_cache	CacheFile
	CacheFile
	UseCanonicalName
	CacheNegotiatedDocs

[mod_cache](#) [mod_cacheURLURL](#) [mod_cache](#)

[mod_proxy](#) [mod_rewrite](#) []

URL [mod_cache](#) Apache

URL [mod_cache](#)(backend)(meta-information)

[UseCanonicalName](#) On (cache key) On
(canonical hostname)

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

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

<!-- -->
<!--#include file="/path/to/footer.html" -->
```

(SSI) virtual

(Expiry Periods)

(3600) CacheDefaultExpire

ExpiresLast-Modified mod_cache

CacheLastModifiedFactor

mod_expires

CacheMaxExpire

(Conditional Request)

(backend)(content provider)Apache(conditional request)

HTTP(header)"Etag:""If-Match:""Last-Modified:""If-Modified-Since:"

"If-Modified-Since:""304 Not Modified"

()

stat()Apache—()

Apache mod_file_cacheApache

mod_cache(cachability)

1. URL CacheEnableCacheDisable
2. HTTP200, 203, 300, 301, 410
3. HTTP GET
4. "Authorization:"
5. "Authorization:""Cache-Control:""s-maxage""must-revalidate""public"

6. URL(GETHTML)"Expires:"RFC261613.9
7. 200(OK) CacheIgnoreNoLastMod"Etag""Last-Modified""Expires"
8. "Cache-Control:""private" CacheStorePrivate
9. "Cache-Control:""no-store" CacheStoreNoStore
10. "Vary:"""*")

HTTP[In short, any content which is highly time-sensitive, or which varies depending on the particulars of the request that are not covered by HTTP negotiation, should not be cached.]

IP5

HTTP"Vary"

/

mod_cache"Vary"

mod_cache"Vary"

"Vary"

Vary: negotiate, accept-language, accept-charset

mod_cacheaccept-languageaccept-charset



(Authorisation)(Access & Control)

mod_cache(reverse-proxy)Apache

.htaccess() mod_cache(authorised) mod_cache

IP CacheDisablemod_expires mod_cacheIP

(Local exploits)

ApacheApache

ApacheCGI mod_disk_cache

Apache mod_disk_cache Apache suEXECApacheCGI

(Cache Poisoning)

Apache''''''''

ApacheDNSDNSApacheHTTP(request-smuggling)

HTTP(google)web



(File Handle Caching)

<u>mod file cache</u>	<u>CacheFile</u>
<u>mod mem cache</u>	<u>CacheEnable</u>
	<u>CacheDisable</u>

ApacheApache

(CacheFile)

Apachemod file cache(file-handle)

CacheFile

CacheFileApache

```
CacheFile /usr/local/apache2/htdocs/index.html
```

CacheFileApacheApache

ApacheApacheApacheApache

CacheEnable fd

mod mem cache CacheEnable

```
CacheEnable fd /
```

mod cache



<u>mod mem cache</u>	<u>CacheEnable</u>
<u>mod file cache</u>	<u>CacheDisable</u>
	<u>MMapStatic</u>

Apacheswap(/)

Linux

```
colm@coroebus:~$ time cat testfile > /dev/null
real    0m0.065s
user    0m0.000s
sys     0m0.001s
colm@coroebus:~$ time cat testfile > /dev/null
real    0m0.003s
user    0m0.003s
sys     0m0.000s
```

""Apache

ApacheApache

Apache

ApacheApacheApache

MMapStatic

mod file cacheMMapStaticApache(mmap())Apache

```
MMapStatic /usr/local/apache2/htdocs/index.html
```

[CacheFileApache](#)

[MMapStaticApache](#)

mod_mem_cache

[mod_mem_cacheHTTP](#) *MMap* [mod_mem_cache](#)

```
#  
CacheEnable mem /  
  
# 1 MB  
MCacheSize 1024
```



(Cache-Store)

<u>mod_disk_cache</u>	<u>CacheEnable</u>
	<u>CacheDisable</u>

mod_disk_cache mod_cache mod_mem_cache

```
CacheRoot /var/cache/apache/  
CacheEnable disk /  
CacheDirLevels 2  
CacheDirLength 1
```

(Cache-Store)

mod_disk_cache URL2URL CGIURL

226422^64URL xyTGxSM02b68mBCykqkp1w URL

CacheDirLevels CacheDirLength

CacheDirLevels CacheDirLength

/var/cache/apache/x/y/TGxSM02b68mBCykqkp1w

CacheDirLength"1"64"2"64*64"1"

CacheDirLength

CacheDirLevels"2"4096100245URL

URLURL(meta-information)".header"".data"

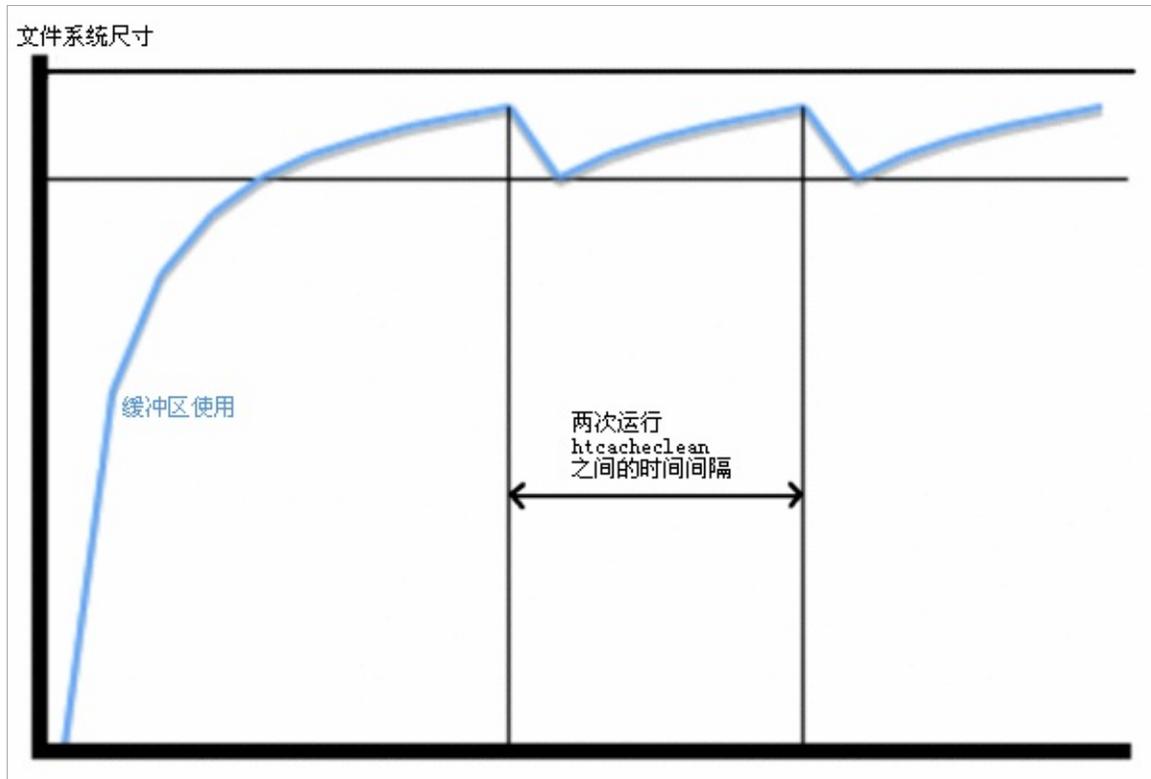
URL

"Vary"URL".vary"".data"

mod_disk_cache

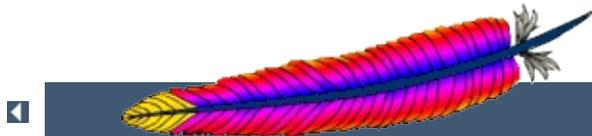
Apache `htcacheclean` `htcacheclean`

`htcacheclean`cron `htcacheclean(G)cron`



1:

mod_disk_cache `htcacheclean`""



Apache HTTP Server 2.2

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

| | | 200616 |

(core)



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

ServerAdminServerTokens() ServerTokensHTTP

ServerNameUseCanonicalNameUseCanonicalPhysicalPort
URL"/"Apache"/"



	<u>CoreDumpDirectory</u>
	<u>DocumentRoot</u>
	<u>ErrorLog</u>
	<u>LockFile</u>
	<u>PidFile</u>
	<u>ScoreBoardFile</u>
	<u>ServerRoot</u>

Apache(/)

ServerRootroot

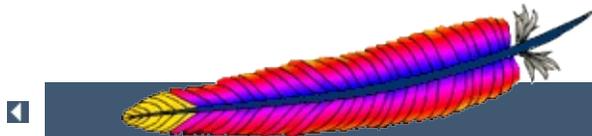


	<u>LimitRequestBody</u>
	<u>LimitRequestFields</u>
	<u>LimitRequestFieldsize</u>
	<u>LimitRequestLine</u>
	<u>RLimitCPU</u>
	<u>RLimitMEM</u>
	<u>RLimitNPROC</u>
	<u>ThreadStackSize</u>

LimitRequest* Apache(DOS)

RLimit* ApacheCGISSI exec

ThreadStackSize



◀ Apache HTTP Server 2.2

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

| | | 200614 |



WebApache HTTP



ApacheApache(root)



	ErrorLog
	LogLevel

[ErrorLog](#)Apache httpd

(unixerror_log WindowsOS/2error.log)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
```

[LogLevel](#)IPWeb

CGI stderr

[\(access log\)](#)403

unix

```
tail -f error_log
```



(Access Log)

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

CustomLog LogFormat

Web Open DirectoryYahoo

Apache httpdmod_log_referer, mod_log_agent TransferLog
CustomLog

Cprintf() mod_log_config

(Common Log Format)

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

```
common "%("                ")"                \n" " \t"
```

CustomLog ServerRoot

(CLF)Web

```
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 HostnameLookups On IPIP logresolve

IPIPIP

- (%1)

identdRFC1413(identity)"-"
On Apache

IdentityCheck

frank (%u)

HTTP(userid) REMOTE_USERCGI401"

-"

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

[//:::]

= 2

= 3

= 4

= 2

= 2

= 2

= (+|-)4

%{format}t formatCstrftime()

"GET /apache_pb.gif HTTP/1.0" ("%r")

GET /apache_pb.gif HTTP/1.0 "

%m

%U%q %H"

%r"

200 (%>s)

(2)(3)(4)(5)

HTTP(RFC261610)

2326 (%b)

" -"" 0" %B

(Combined Log Format)

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

```
CustomLog log/access_log combined
```

```
%{header}i header
```

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

CustomLogCLF

CustomLogReferLogAgentLog

```
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"
```

CustomLog LogFormat

SetEnvIf CustomLog env=

```
#  
SetEnvIf Remote_Addr "127\.0\.0\.1" dontlog  
# robots.txt
```

```
SetEnvIf Request_URI "^/robots\.txt$" dontlog
#
CustomLog logs/access_log common env=!dontlog
```

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



100001MBApache

(graceful)

```
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(")

Apache httpdroot

[rotatelogs](#) 24

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

[cronolog](#)



<VirtualHost>

CustomLogErrorLog<VirtualHost>

<VirtualHost>

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

%v [split-logfile](#)



mod_logio	LogFormat
mod_log_forensic	ForensicLog
mod_cgi	PidFile
mod_rewrite	RewriteLog
	RewriteLogLevel
	ScriptLog
	ScriptLogBuffer
	ScriptLogLength

[mod_logio](#)[LogFormat](#)(%l %O)

(Forensic Logging)

[mod_log_forensic](#)(forensic log)(forensic logger)

PID

Apache httpd logs/httpd.pidhttpdID(process id[PID])

[PidFile](#)PIDWindows -k

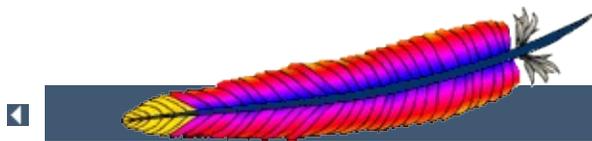
[ScriptLog](#)CGI

[mod_cgi](#)

[mod_rewrite](#)

[RewriteLog](#)

[RewriteLogLevel](#)



Apache HTTP Server 2.2

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

| | | 200617 |

URL

ApacheURL



<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>ProxyPassReverseCookieDomain</u>
	<u>ProxyPassReverseCookiePath</u>
	<u>Redirect</u>
	<u>RedirectMatch</u>
	<u>RewriteCond</u>
	<u>RewriteMatch</u>
	<u>ScriptAlias</u>
	<u>ScriptAliasMatch</u>
	<u>UserDir</u>



ApacheURL(URL) DocumentRoot

Apache DocumentRoot mod_vhost_aliasIP



DocumentRootApacheUnix DocumentRoot
OptionsFollowSymLinksSymLinksIfOwnerMatch

Alias

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

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

AliasMatchScriptAliasMatch

```
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_userdirURL
```

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

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

```
/etc/passwd Userdir
```

```
"~"(%7e) mod_userdir AliasMatch  
http://www.example.com/upages/user/file.html  
/home/user/public_html/file.html AliasMatch
```

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



ApacheURLURL (redirection)Redirect
DocumentRoot/foo//bar/

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

/foo/URLwww.example.com/bar/

ApacheRedirectMatch

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

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



ApacheWeb() (reverse proxying)

/foo/ internal.example.com/bar/

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

ProxyPass ProxyPassReverseinternal.example.com
ProxyPassReverseCookieDomain
ProxyPassReverseCookieDomaincookie

internal.example.com mod_proxy_htmlHTML
XHTML



mod_rewriteURLIP(aliases)

URL



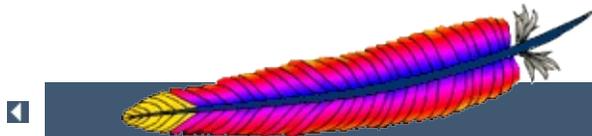
URL [URL](#)

HTMLURLApache [mod_speling](#)"File Not Found"

[mod_speling](#)URLunixURL""URL

Apache"404"() [ErrorDocument](#)

||||



Apache HTTP Server 2.2

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

| | | 2006110 |

Apache



Apache HTTP Apache Apache HTTP
Apache

[Apache HTTP](#)

WebApacheCGI



Apacheroot Userroot ServerRootrootroot
 ServerRoot/usr/local/apache root

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

"/usr"/usr/local"root [httpd](#)

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

htdocs -- root

rootroot [httpd](#)(root)(root)



(SSI)

ApacheSSISSI

SSICGI"exe ccmd"SSICGIhttpd.confApache

SSISSI

[CGIsuexec](#) SSI

.html.htmSSISSI.shtml

```
SSI Options IncludesNOEXECIncludes <--  
#include virtual="..." --> ScriptAliasCGI
```



CGICGICGIweb

CGI()ABB

[CGIWrap](#)

[suEXEC](#)Apache1.2Apache



CGI

-
-
-



CGICGICGI/

CGI



Apachemod_php, mod_perl, mod_tcl, mod_python Apache(
User)Apache



.htaccess

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

.htaccess



ApacheURL

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

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

Directory

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

LocationDirectory <Directory /> <Location />

UserDir"/"1.3

```
UserDir disabled root
```



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

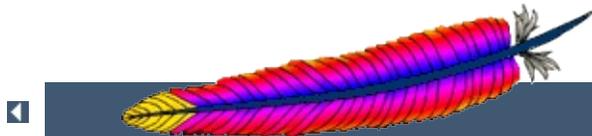
[Apache Tomcat Source.JSP Malformed Request Information Disclosure Vulnerability](#)

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

.htpasswd

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

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



Apache HTTP Server 2.2

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

| | | 200612 |

(DSO)

Apache HTTP [httpd](#) [httpd\(DSO\)DSO](#)Apache
([apxs](#))

DSO



<code>mod_so</code>	<code>LoadModule</code>

ApacheDSOApache `mod_so` `core`DSOApache `--enable-`
`module=shared` DSO `mod_foo.so`DSO `httpd.conf`
`mod_so``LoadModule`

`apxs`(*APache eXtenSion*)ApacheDSOApache DSOApache
`configure` `make` `install` Apache C `apxs`Apache
DSO



Apache2.0 DSO

1. Apache mod_foo.c mod_foo.so DSO

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

2. mod_foo.c mod_foo.so DSO

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

3. Apache

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

4. [apxs](#) Apache mod_foo.c mod_foo.so DSO

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

httpd.conf [LoadModule](#) Apache



Unix(DSO)/

ld.soUnix dlopen()/dlsym()

DSO (shared libraries)DSO(DSO libraries) libfoo.so
libfoo.so.1.2(/usr/lib) -lfoo -R
LD_LIBRARY_PATH Unix /usr/liblibfoo.so DSO

DSO()DSOUnix(ld.so) libc.so

DSO (shared objects) DSO(DSO files)(foo.so)
dlopen()DSODSOUnixDSODSO(libc.so)DSO

DSO API dlsym()DSO ()

DSODSO()DSO""DSO()DSO
DSO

DSO

1998DSO Perl 5(XSDynaLoader)Netscape Server
1.3ApacheApache(dispatch-list-based)ApacheApacheDSO

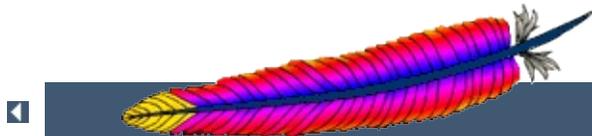


DSO

- `httpd.conf` [LoadModule](#) Apache(&SSL& [mod_perlPHP])
- ApachePHPmod_perlmod_fastcgi
- ApacheDSO [apxs](#)Apache `apxs -i apachectl restart Apache`

DSO

- DSO
- Unix20%
- (position independent code[PIC])5%
- DSODSO(ld -lfoo)a.outELFDSODSO
ApacheC(libc)Apache(libfoo.a)Apache
dlopen()



Apache HTTP Server 2.2

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

| | | 200612 |

ApacheHTTP/1.1

mod_negotiation



Accept-Language: fr

HTMLGIFJPEG

```
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
```

ApacheHTTP/1.1" AcceptAccept - LanguageAccept -
CharsetAccept - EncodingRFC2295RFC2296RFC"

(resource)URI(RFC2396)HTTPApache **(representation)**



- (*.var)
- "MultiViews"

```
type-map(Apache MIMEapplication/x-type-map)  
type-map
```

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

```
(entry)HTTP()                foofoo.var
```

```
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 On "qs" jpeg, gif, ASCII-art
```

```
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
```

```
qs0.0001.0000.000qsqs1.0qs""jpegASCII-art
jpegqs
```

[mod_negotiation](#)HTTP

Multiviews

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

```
MultiViews /some/dir/foo /some/dir/foo
/some/dirMultiViews foo.* foo.*
```

```
MultiViews DirectoryIndex
```

```
DirectoryIndex index
```

```
index.htmlindex.html3 index.cgi
```

```
mod_mime MultiViewsMatchMultiViews
```



Apache""Apache

1. **Apache** ()Apache""(dimension)
2. RFC2295""ApacheRFC2296""

(Dimension)

Accept("qs")
Accept - Language
Accept - Encoding
Accept - Charset

Apache

Apache""

1. *Accept** *Accept** 4
2. ""3
 1. Accept
 - 2.
 3. Accept - Language() LanguagePriority()
 4. ""(text/html)
 5. Accept - CharsetISO-8859-1 text/* ISO-8859-1
 6. ISO-8859-1
 - 7.
 - 8.

9. ASCII

3. ""HTTP Vary()
4. ()406HTMLHTTP Vary



ApacheApache

Accept

```
Accept: ""image/*""/*"
```

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

```
"image/"("image/*")
```

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

```
""/*""*.*)"()0.01
```

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

```
1.0""/*""0.01
```

```
Accept: qApache""/*""q0.01"type/*"q0.02""/*"
```

```
Accept: q
```

Apache 2.0

```
Accept - language "No Acceptable Variant" "Multiple  
Choices" Apache Accept - languag  
ForceLanguagePriority LanguagePriority
```

```
en-GBHTTP/1.1 en ( Accept - Languageen-GBen  
)"No Acceptable Variants"  
LanguagePriorityApache"en-GB; q=0.9, fr; q=0.8"
```

"fr"HTTP/1.1

(cookiesURL)2.0.47 mod_negotiationprefer-language
mod_negotiation

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



Apache{encoding ..}(RFC2295)RVSA/1.0(RFC2296)
Accept-EncodingRVSA/1.0



([mod_mime](#))

MIME(html)(gz)(en)

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

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

(foo)rsp. htmlshtmlcgi

MIME(foo.html)()MIME(foo.html.en)

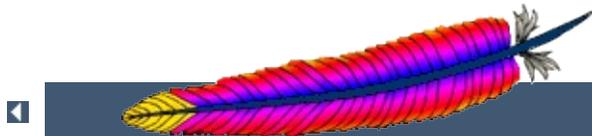


URL(representation)URLApacheHTTP/1.1
ApacheHTTP/1.1

HTTP/1.0() CacheNegotiatedDocsHTTP/1.1

HTTP/1.1Apache Vary force-no-vary





Apache HTTP Server 2.2

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

| | | 200612 |



Apache

"500 Server Error"URL()



Apache1.3

- 1.
2. URL
3. URL

URL/

ApacheCGI

```
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_URLREDIRECT_QUERY_STRINGURL(URLcgicgi)
[ErrorDocument](#)(http:)



ErrorDocument .htaccessAllowOverride

...

```
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> <action>
```

<action>

1. ("
2. URL
3. URL



ApacheURL/

CGI

```
" REDIRECT_ REDIRECT_* CGI" REDIRECT_  
HTTP_USER_AGENTREDIRECT_HTTP_USER_AGENT Apache  
REDIRECT_URLREDIRECT_STATUSURLURL
```

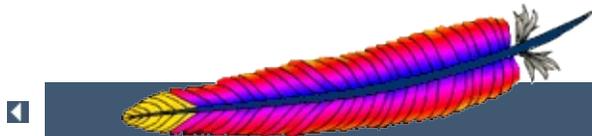
```
ErrorDocumentCGI" Status:"Perl ErrorDocument
```

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

404 Not Found

```
" Location:()" " Status: "( 302 Found)  
" Location:"
```

||||



Apache HTTP Server 2.2

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

| | | 200611 |

(Binding)

Apache



<u>core</u>	<u><VirtualHost></u>
<u>mpm_common</u>	<u>Listen</u>

ApacheIP()

Listen(+)

ListenIP+

Listen

808000

```
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 ApacheIPv6IPv6

ApacheIPv6IPv4IPv6IPv6IPv4IPv6IPv4(IPv4-mapped IPv6 addresses)FreeBSDNetBSDOpenBSDApache

(LinuxTru64)IPv6IPv4 (mapped addresses)ApacheIPv4IPv6
IPv4IPv6 --enable-v4-mapped

FreeBSDNetBSDOpenBSD --enable-v4-mapped
Apache

ApacheIPv4APR [Listen](#)IPv4

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

IPv6IPv4ApacheIPv4IPv6() --disable-v4-mapped -
-disable-v4-mapped FreeBSDNetBSDOpenBSD



Listen(main server)

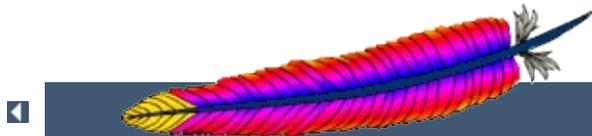
<VirtualHost>

<VirtualHost>

<VirtualHost>

<VirtualHost>

| | | |



Apache HTTP Server 2.2

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

| | | 200615 |



Apache



Apache HTTP Apache

Apache 2.0 web (MPM)

- Apache [mpm_winnt](#) Apache 1.3 POSIX Windows Apache MPM
- [worker](#) [event](#) MPM [prefork](#)

MPM Apache MPM MPM



MPMMPMUnixMPMApacheApache

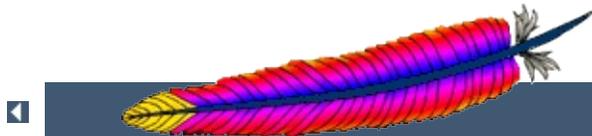
[configure](#) --with-mpm=*NAME* MPM *NAME*MPM

./httpd -l MPMMPM



MPMMPM

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



Apache HTTP Server 2.2

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

| | | 200613 |

Apache

Apache HTTP (environment variable)CGI

ApacheCGI(SSl)shell



<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 [SetEnv](#) [PassEnv](#)Apacheshell

mod_setenvif (User-Agent)"Referer:"
 mod_rewrite[RewriteRule](#) [E=...]

mod_unique_idUNIQUE_ID""

CGI

ApacheshellCGISSI [CGI](#)

- CGI
- [suexec](#)CGICGI suexec.c
- CGISSI



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

CGI

CGICGIApache

[CGI](#)

SSI

mod_includeINCLUDES(Server-parsed[SSI])echoApacheCGI

SSI

[SSI](#)

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

[LogFormat](#)" %e"

[CustomLog](#)

[SetEnvIf](#)

gif

[Header](#)HTTP

```
mod_ext_filter ExtFilterDefine disableenv=  
enableenv=
```

URL

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



Apache

[BrowserMatch](#) [SetEnvPassEnv](#)

downgrade-1.0

HTTP/1.0

force-gzip

DEFLATEaccept-encodinggzip

force-no-vary

Vary [force-response-1.0](#)

force-response-1.0

HTTP/1.0HTTP/1.0AOLHTTP/1.0HTTP/1.1

gzip-only-text/html

```
"1" text/htmlmod\_deflateDEFLATE  
mod\_negotiation(gzip")
```

no-gzip

[mod_deflate](#)DEFLATE [mod_negotiation](#)

nokeepalive

[KeepAlive](#)

prefer-language

[mod_negotiation](#)(enfrzh_cnx-) [mod_negotiation](#)

redirect-carefully

WebFoldersDAV

suppress-error-charset

2.0.54

Apache()ApacheISO-8859-1

Apache

force-proxy-request-1.0, proxy-nokeepalive, proxy-sendchunked, proxy-sendcl

mod_proxy mod_proxy



httpd.conf

```
# HTTP
# Netscape 2.xkeepalive
# IE4.0HTTP/1.1301/302()keepalive
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0

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

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

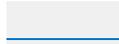
....

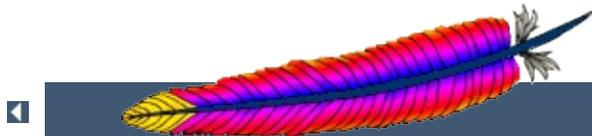
/web/images

```
SetEnvIf Referer "^http://www.example.com/" local_referer
# Referer
SetEnvIf Referer "^$" local_referral
<Directory /web/images>
    Order Deny,Allow
    Deny from all
    Allow from env=local_referral
```

</Directory>

"Apache"





Apache HTTP Server 2.2

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

| | | 200614 |

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_imagemap</u>	<u>SetHandler</u>
<u>mod_info</u>	
<u>mod_mime</u>	
<u>mod_negotiation</u>	
<u>mod_status</u>	

""Apache""

Apache1.1 ()

Action

- **default-handler**default_handler()(core)
- **send-as-is**HTTP(mod_asis)
- **cgi-script**CGI(mod_cgi)
- **imap-file**(mod_imagemap)
- **server-info**(mod_info)
- **server-status**(mod_status)
- **type-map**(mod_negotiation)



CGI

```
htmlCGI footer.pl
```

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

```
CGI(    PATH_TRANSLATED)
```

HTTP

```
send-as-isHTTP    /web/htdocs/asis/    send-  
as-is
```

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



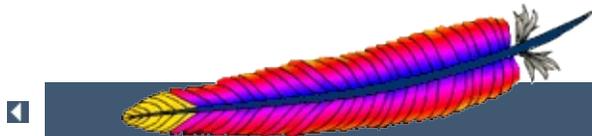
[Apache API](#)

[Apache API](#)

```
char *handler
```

```
invoke_handler r->handler"-"/"
```

| | | |



Apache HTTP Server 2.2

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

| | | 200613 |

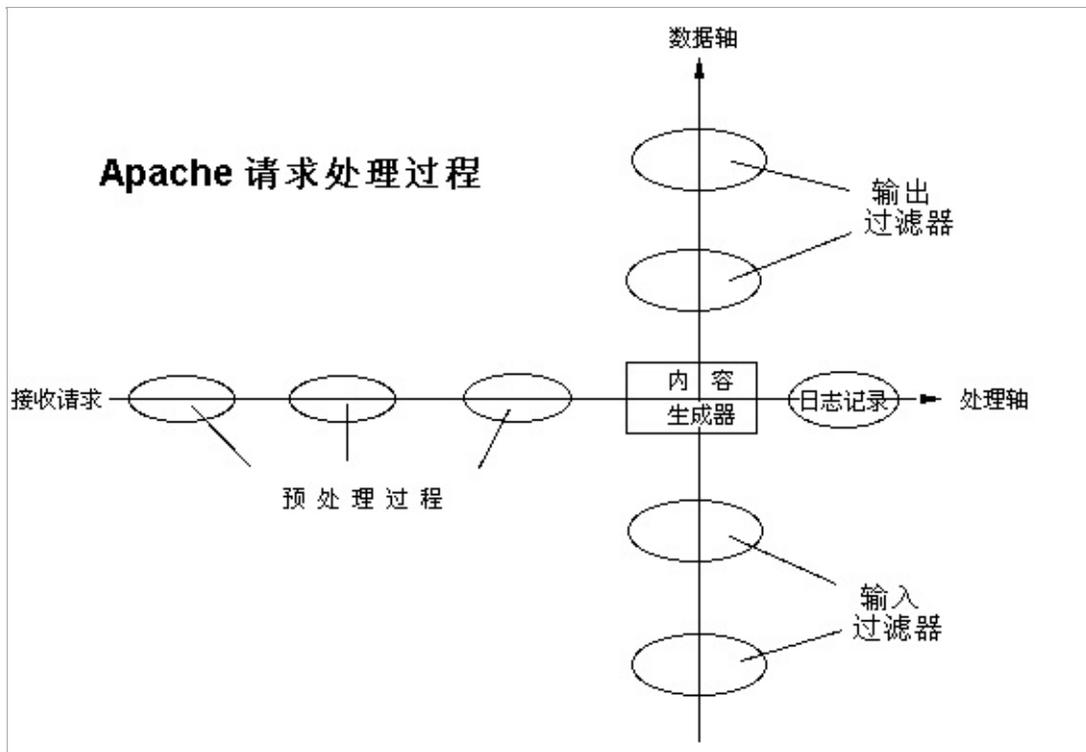
(Filter)

Apache



<u>mod filter</u> <u>mod deflate</u> <u>mod ext filter</u> <u>mod include</u> <u>mod charset_lite</u>	FilterChain FilterDeclare FilterProtocol FilterProvider AddInputFilter AddOutputFilter RemoveInputFilter RemoveOutputFilter ExtFilterDefine ExtFilterOptions SetInputFilter SetOutputFilter
---	--

Apache 2.0 (post-process)



Apache

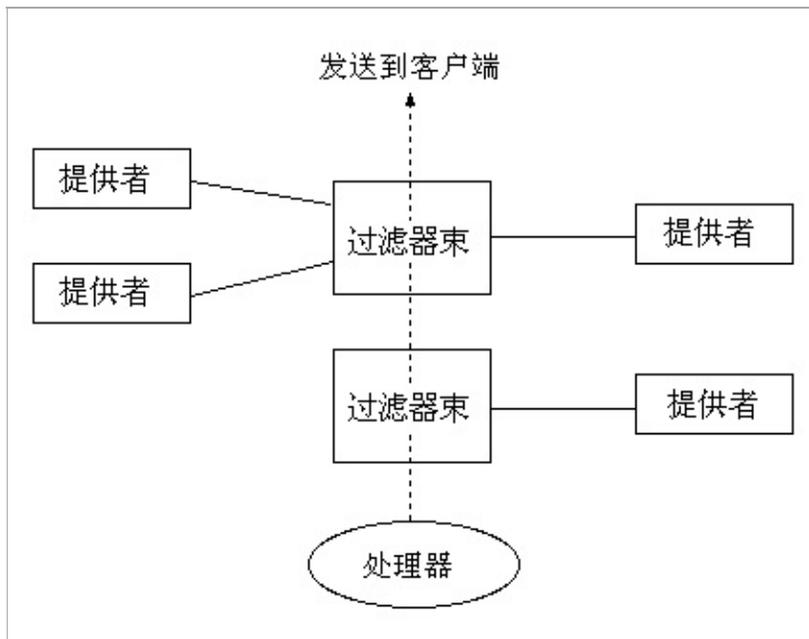
- [mod_include](#)
- [mod_ssl](#)SSL(https)
- [mod_deflate](#)/
- [mod_charset_lite](#)
- [mod_ext_filter](#)

Apache(byte-range handling)

modules.apache.org

- HTMLXML
- XSLTXIncludes
- XML
- HTML
-
- PHP
-





Apache 2.1 [mod_filter](#)HTMLJPEG(filter harness)
(provider)(provider)

- HTMLtext/htmlapplication/xhtml+xml
-
-



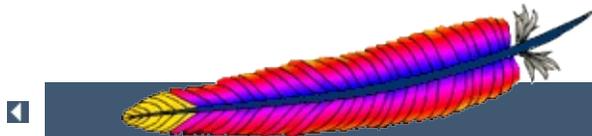
()

AddInputFilter, AddOutputFilter, RemoveInputFilter,
RemoveOutputFilter

mod filter FilterChain, FilterDeclare,
FilterProvider

AddOutputFilterByType

|||



Apache HTTP Server 2.2

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

| | | 200616 |

suEXEC

suEXECApacheWebCGISSICGISSIweb

CGISSI

setuid root suEXEC



Apache

UNIX **setuidsetgid**suEXEC

setuid/setgid

suEXECsuEXEC

Apache

Apache suEXECsuEXECsuEXECsuEXEC

ApachesuEXEC

suEXEC



suEXECsuEXEC

suEXECsetuid""Apache webHTTP""CGI
SSIApacheUIDGIDsuEXEC

(wrapper)("""CGI/SSI")

1.

2.

Apache webApachesuEXEC

3.

(Apache)

4. **CGI/SSI**

CGI/SSI"/"".."suEXEC(--with-
suexec-docroot=*DIR*)

5.

6.

7.

suEXECrootCGI/SSI

8. **UIDUID**

UIDCGI/SSIUID

9.

suEXECrootCGI/SSI

10. **GIDGID**

GIDCGI/SSIGID

11.

setuidsetgid

12.

13. **Apache**

suEXECsuEXEC([suEXEC](#))

14.

15. **CGI/SSI**

16. **CGI/SSI**

17. **setuidsetgid**

UID/GID

18.

19.

suEXEC()

20.

suEXEC

suEXEC CGI/SSI

suEXEC



...

suEXEC

--enable-suexec

suEXEC --with-suexec-xxxxx APACIsuEXEC

--with-suexec-bin=PATH

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

--with-suexec-caller=UID

ApacheUID

--with-suexec-userdir=DIR

suEXECsuEXEC"" UserDir("*")
UserDir"passwd"suEXEC"public_html" UserDir
"~userdir"cgi

--with-suexec-docroot=DIR

ApacheDocumentRootUserDirsuEXEC --datadir
"/htdocs"" --datadir=/home/apache"
"/home/apache/htdocs"suEXEC

--with-suexec-uidmin=UID

suEXECUID500100100

--with-suexec-gidmin=GID

suEXECGID100100

--with-suexec-logfile=FILE

suEXEC()"suexec_log"(--logfiledir)

--with-suexec-safepath=PATH

CGIPATH"/usr/local/bin:/usr/bin:/bin"

suEXEC

--enable-suexec suEXEC make(Apache) suexec
make install suexec --sbindir
"/usr/local/apache2/sbin/suexec"

```
rootsuEXECUID root 1)
```

```
suEXEC --with-suexec-caller suEXEC
```

```
ApachesuEXEC
```

```
web-server
```

```
User www  
Group webgroup
```

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

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

```
ApachesuEXEC
```



Apache --sbindir ("/usr/local/apache/sbin/suexec")
[suexec](#)ApachesuEXEC

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

setuid root

ApachesuEXECApacheHUPUSR1

suEXEC [suexec](#)Apache



CGIsuEXEC SuexecUserGroup mod_userdir

suEXECVirtualHostSuexecUserGroupUIDCGI
<VirtualHost>UserGroup <VirtualHost>UID

mod_userdirsuEXECUIDCGICGI --with-
suexec-userdir



suEXEC --with-suexec-logfile



Apache

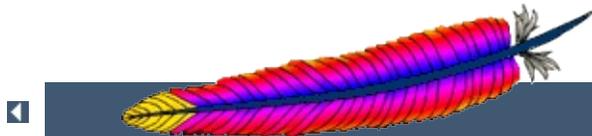
suEXEC"bugs"

- **suEXEC**
-

suEXEC4ApachesuEXEC()

- suEXECPATH

- suEXEC



Apache HTTP Server 2.2

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

| | | 2006321 |



Apache2.0webApache2.0

Apache 1.3 2.0Apache2.0httpd



webweb''''''''''

MaxClients

topApache

CPU''''

- TCP
- `sendfile()`(LinuxLinux2.4Solaris8)
sendfileApache2CPU



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

HostnameLookups DNS

Apache1.3 HostnameLookups On DNSApache1.3

Off logresolve DNS

web

" Allow from domain"" Deny from domain"(domainIP)DNS
)(IP)

<Location /server-status>DNS .html.cgiDNS

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

CGIDNS gethostbyname

FollowSymLinks SymLinksIfOwnerMatch

Options FollowSymLinks Options
SymLinksIfOwnerMatch Apache

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

```
" /index.html"Apache  
" /www"" /www/htdocs"" /www/htdocs/index.html"  
lstat() lstat()
```

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

DocumentRoot

AliasRewriteRuleDocumentRoot

FollowSymLinks

AllowOverride

(.htaccess)Apache .htaccess

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

```
" /index.html"Apache  
" /.htaccess"" /www/.htaccess"" /www/htdocs/.htaccess"
```

Options FollowSymLinks

AllowOverri

```
DirectoryIndex index
```

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

```
type-map" Options MultiViews" type-map
```

Apache2.0 mmap()

httpd

- CPU mmapread()Solaris mmap Apache2.0
- NFSNFS

EnableMMAP off

Sendfile

Apache2.0() sendfile() Apachesendfile()

sendfilesendfilehttpd

- Apachesendfilesendfile

- NFScache

" EnableSendfile off"sendfile

Apache1.3 MinSpareServers, MaxSpareServers,
StartServersApache"" StartServers
MinSpareServers100 StartServers595
10

""Apache1.3""32
MinSpareServers

MinSpareServers, MaxSpareServers, StartServers 4
ErrorLog mod_status

MaxRequestsPerChild" 0"30SunOSSolaris
10000

KeepAliveTimeout5 60 most of the benefits are
lost



MPM

Apache 2.x [\(MPM\)](#)ApacheMPMUNIXMPM [beos](#),
[mpm_netware](#), [mpmt_os2](#), [mpm_winnt](#)UNIXMPMhttpd

- [worker](#)MPMMPM [prefork](#)MPM
- [prefork](#)MPM [worker](#)MPM [worker](#)MPM(
php3/4/5) [worker](#)MPM

MPM

[DSO](#) [LoadModule](#)

ApacheApache

[mod_mime](#), [mod_dir](#), [mod_log_config](#)
mod_log_config

[mod_cacheworker](#)APR(Apache)APIAPI

APROS/CPUCPU(compare-and-swap, CAS)APRAPI
CASCPUAPUApache

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

--enable-nonportable-atomics

- SPARCSolaris

```

APR                                     --enable-nonport
SPARC v8plusCASUltraSPARC CPU
• x86Linux
APRLinux                               --enable-i
APR486CAS486CPU

```

mod_status "ExtendedStatus On"

```

Apachemod_status " ExtendedStatus On"Apache
gettimeofday( times())(1.3) time()
" ExtendedStatus off"()

```

socketaccept

Apache2.0

Unix socket API web [ListenApache](#) select()socket
select()socketApache()

```

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;
}

```

```

""          select accept()          accept socket
""
socket CPU          select109          accept  select
socket                                     selectCPU

```

Apache()

```

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

        FD_ZERO (&accept_fds);
        for (i = first_socket; i <= last_socket; ++i)
        {
            FD_SET (i, &accept_fds);
        }
        rc = select (last_socket+1, &accept_fds,
            NULL, NULL, NULL);
        if (rc < 1) continue;
        new_connection = -1;
        for (i = first_socket; i <= last_socket; ++i)
        {
            if (FD_ISSET (i, &accept_fds)) {

```

```

        new_connection = accept (i, NULL, NULL);
        if (new_connection != -1) break;
    }
}
if (new_connection != -1) break;
}
accept_mutex_off ();
process the new_connection;
}

```

accept_mutex_onaccept_mutex_off src/conf.h(1.3
) src/include/ap_config.h(1.3) [Listen](#)

[AcceptMutex](#)

AcceptMutex flock

flock()([LockFile](#))

AcceptMutex fcntl

fcntl()([LockFile](#))

AcceptMutex sysvsem

(1.3)SysV SysV Apache(ipcs() man page)
APIuidCGI(CGI)

AcceptMutex pthread

(1.3)POSIX POSIX Solaris2.5

AcceptMutex posixsem

(2.0)POSIX segfault

APR(Apache)

[Listen](#)

socketaccept

socketsocket accept()""TCP
acceptsocket

socketLinux(2.0.30Pentium pro 166/128M
RAM)socket3%100msLANsocket
SINGLE_LISTEN_UNSERIALIZED_ACCEPT socket

[draft-ietf-http-connection-00.txt](#) section 8HTTP (TCP
)1.2Apache

UnixTCP FIN_WAIT_2Apache1.2 FIN_WAIT_2
TCP/IP(SunOS4 --)

socket SO_LINGER TCP/IP(Linux2.0.31)

Apachelingering_close(http_main.c)

```
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);
}
```

HTTP/1.1 NO_LINGCLOSE HTTP/1.1
lingering_close

Scoreboard

```
Apachescoreboard() src/main/conf.h
USE_MMAP_SCOREBOARDUSE_SHMGET_SCOREBOARD (
HAVE_MMAPHAVE_SHMGET)
()
```

LinuxApache1.2ApacheLinux

DYNAMIC_MODULE_LIMIT

```
() -DDYNAMIC_MODULE_LIMIT=0
```



Solaris8MPMApache2.0.38

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

-l trussLWP(lightweight process--Solaris)ID

strace, ktrace, par

httpd10KB()

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

LWP #67

```
accept()MPMaccept
```

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

LWP #65

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

Apachesocket([Listen](#))

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

brk()httpd(apr_poolapr_bucket_alloc)httpd
malloc()

```
/65: fcntl(9, F_GETFL, 0x00000000)
/65: fstat64(9, 0xFAF7B818)
/65: getsockopt(9, 65535, 8192, 0xFAF7B918, 0xFAF7B918)
/65: fstat64(9, 0xFAF7B818)
/65: getsockopt(9, 65535, 8192, 0xFAF7B918, 0xFAF7B918)
/65: setsockopt(9, 65535, 8192, 0xFAF7B918, 4, 219)
/65: fcntl(9, F_SETFL, 0x00000082)
```

setsockopt()getsockopt()Solarislibcsocketfcntl()

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

```
/65: stat("/var/httpd/apache/httpd-8999/htdocs/10k
/65: open("/var/httpd/apache/httpd-8999/htdocs/10k
```

```
httpd" Options FollowSymLinks" AllowOverride
None" lstat().htaccess stat()
```

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

```
httpd sendfilev()HTTPSendfile sendfile()
write()writev()
```

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

```
write() time()Apache1.3Apache2.0
gettimeofday()LinuxSolaris gettimeofday
```

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

```
/65: close(9)
```

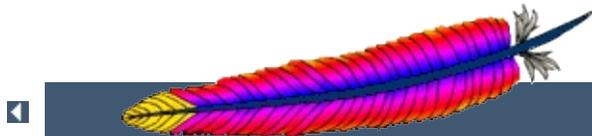
```
/65: close(10)
```

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

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

```
(MPM) accept()
```

```
||||
```



Apache HTTP Server 2.2

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

| | | 2006321 |

URL

Originally written by
Ralf S. Engelschall <rse@apache.org>
December 1997

[mod_rewrite](#)URLURL



Apache mod_rewrite URL
mod_rewrite mod_rewrite

mod_rewrite Apache



URL

```
mod_aliasmod userdir[PT].htaccess
```



URL

webURLURLURLURL

URLHTTP/u/user/~user/u/user

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

www.example.com

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

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

DocumentRoot

web [DocumentRoot](#)URL/"Intranet
/e/www/(WWW)/e/sww/(Intranet)
/e/www/

[DocumentRoot](#)

URL"/"/e/www/"mod_rewriteURL Aliases(
mod_alias)DocumentRootURLmod_rewrite

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

RedirectMatch

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

/~quux/foo/~quux/foo/ foo
CGIURL

URL/~quux/foo/index.html
image.gif/~quux/image.gif

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

.htaccess

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

URL

IntranetWWWURLURL()WWWURL

()

```
user1  server_of_user1
user2  server_of_user2
:      :
```

map.xxx-to-hostURLURL

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

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

(server0)

RewriteEngine on

```
RewriteMap      user-to-host      txt:/path/to/map.us
RewriteMap      group-to-host     txt:/path/to/map.gr
RewriteMap      entity-to-host    txt:/path/to/map.en

RewriteRule     ^/u/([^/]+)/?(.*) http://${user-to
RewriteRule     ^/g/([^/]+)/?(.*) http://${group-to
```

```

RewriteRule ^/e/([^/]+)/?(.*) http://${entity-to
RewriteRule ^/([uge])/([^/]+)/?$ /$1/$2
RewriteRule ^/([uge])/([^/]+)/([^.]+.+)$ /$1/$2

```

web

webwebweb

webURL"/~user/anypath"http://newserver/~user/anypath

```

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

```

```

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

```

~

```

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

```

net.sw1992Unix

```

drwxrwxr-x  2 netsw  users    512 Aug  3 18:39 Au
drwxrwxr-x  2 netsw  users    512 Jul  9 14:37 Be

```

```

drwxrwxr-x 12 netsw users 512 Jul 9 00:34 Cr
drwxrwxr-x 5 netsw users 512 Jul 9 00:41 Da
drwxrwxr-x 4 netsw users 512 Jul 30 19:25 Di
drwxrwxr-x 10 netsw users 512 Jul 9 01:54 Gr
drwxrwxr-x 5 netsw users 512 Jul 9 01:58 Ha
drwxrwxr-x 8 netsw users 512 Jul 9 03:19 In
drwxrwxr-x 3 netsw users 512 Jul 9 03:21 Ma
drwxrwxr-x 3 netsw users 512 Jul 9 03:24 Mi
drwxrwxr-x 9 netsw users 512 Aug 1 16:33 Ne
drwxrwxr-x 2 netsw users 512 Jul 9 05:53 Of
drwxrwxr-x 7 netsw users 512 Jul 9 09:24 So
drwxrwxr-x 7 netsw users 512 Jul 9 12:17 Sy
drwxrwxr-x 12 netsw users 512 Aug 3 20:15 Ty
drwxrwxr-x 10 netsw users 512 Jul 9 14:08 X1

```

19967Web""CGIFTPWebCGI

CGI/e/netsw/.www/

```

-rw-r--r-- 1 netsw users 1318 Aug 1 18:10 .
drwxr-xr-x 18 netsw users 512 Aug 5 15:51 D
-rw-rw-rw- 1 netsw users 372982 Aug 5 16:35 L
-rw-r--r-- 1 netsw users 659 Aug 4 09:27 T
-rw-r--r-- 1 netsw users 5697 Aug 1 18:01 n
-rwxr-xr-x 1 netsw users 579 Aug 2 10:33 n
-rwxr-xr-x 1 netsw users 1532 Aug 1 17:35 n
-rwxr-xr-x 1 netsw users 2866 Aug 5 14:49 n
drwxr-xr-x 2 netsw users 512 Jul 8 23:47 n
-rwxr-xr-x 1 netsw users 24050 Aug 5 15:49 n
-rwxr-xr-x 1 netsw users 1589 Aug 3 18:43 n
-rwxr-xr-x 1 netsw users 1885 Aug 1 17:41 n
-rw-r--r-- 1 netsw users 234 Jul 30 16:35 n

```

"DATA"net.sw rdistURLCGIURL"DATA"
DocumentRootURL"/net.sw/""/e/netsw"

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

[/e/netsw/.www/.wwwacl](#)

```
Options ExecCGI FollowSymLinks Includes Mult
RewriteEngine on

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

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

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

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

# cgi
RewriteRule !^netsw-lsdir\.cgi.* -
RewriteRule (.*) netsw-lsdir
```

1. L()("-")
2. !()C()

3.

NCSAmod_imap

NCSA webApache webNCSAApache [mod_imagemap](#)
/cgi-bin/imagemap/path/to/page.mapimagemapApache
/path/to/page.map

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

webMultiViews

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

URL

CGIURL

XSSICGI "/foo/S=java/bar/"URL/foo/bar/ STATUS
"java"

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

usernamewww.**username**.host.domain.comDNS

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

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

ourdomain.comURLwebwww.somewhere.com

```
RewriteEngine on
```

```
RewriteCond %{REMOTE_HOST} !^.+\.ourdomain\.com
RewriteRule ^(/~.+)$ http://www.somewhere
```

URLweb

[URLwebABPerlCGI](#) [ErrorDocument](#)
[mod_rewrite](#) [ErrorDocumentCGI!](#)

```
RewriteEngine on
RewriteCond /your/docroot/%{REQUEST_FILENAME} !-
RewriteRule ^(.+)$ ht
```

[DocumentRoot\(\)](#)

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

[mod_rewrite](#)""(look-ahead)URLwebweb

CPU

[Error](#)

URL()ApacheURLuri_escape()(anchor)

"url#anchor"URL [mod_rewrite](#)URL?

NPH-CGINPH(HTTP)()URL"xredirect:"

```
RewriteRule ^xredirect:(.+)$ /path/to/nph-xredirect
```

```
[T=application/x-httpd-cgi,L]
```

```
"xredirect:"URLnph-xredirect.cgi
```

```
#!/path/to/perl
##
##  nph-xredirect.cgi -- NPH/CGI script for extend
##

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

print "HTTP/1.0 302 Moved Temporarily\n";
print "Server: $ENV{'SERVER_SOFTWARE'}\n";
print "Location: $url\n";
print "Content-type: text/html\n";
print "\n";
print "<html>\n";
print "<head>\n";
print "<title>302 Moved Temporarily (EXTENDED)</ti
print "</head>\n";
print "<body>\n";
print "<h1>Moved Temporarily (EXTENDED)</h1>\n";
print "The document has moved <a HREF=\""$url\">her
print "</body>\n";
print "</html>\n";

##EOF##
```

```
URL mod_rewrite"news:newsgroup"
```

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

```
[R][R,L]"xredirect:""
```

<http://www.perl.com/CPAN>CPAN(Perl)CPANFTPFTP
CPANCGI mod_rewrite

mod_rewrite3.0.0"ftp:" RewriteMap

```
RewriteEngine on
RewriteMap    multiplex          txt:/path/t
RewriteRule   ^/CxAN/(.*)       %{REMOTE_HO
RewriteRule   ^.+\.([a-zA-Z]+)::(.*)$  ${multiplex
```

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

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

CGI mod_rewrite

TIME_xxx"<STRING", ">STRING""=STRING"

```
RewriteEngine on
RewriteCond   %{TIME_HOUR}%{TIME_MIN} >0700
```

```
RewriteCond    %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule    ^foo\.html$             foo.day.html
RewriteRule    ^foo\.html$             foo.night.ht
```

URLfoo.html07:00-19:00foo.day.html foo.night.html ...

YYYYXXXX

.html.phtml .YYYY.XXXXURL()

```
# 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$             $1           [C
# rewrite to document.phtml if exists
RewriteCond    %{REQUEST_FILENAME}.phtml -f
RewriteRule    ^(.*)$ $1.phtml          [S
# else reverse the previous basename cutout
RewriteCond    %{ENV:wasHTML}           ^yes$
RewriteRule    ^(.*)$ $1.html
```



URL()

:
bar.htmlfoo.html URLURL

:
URL

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

URL()

:
bar.htmlfoo.html URLURL

:
HTTP

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

:
NetscapeLynx

:
HTTP"User-Agent"HTTP"User-
Agent""Mozilla/3" foo.htmlfoo.NS.html "Lynx"
12"Mozilla" foo.20.html foo.32.html

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

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

RewriteRule ^foo\.html$ foo.32.html
```

[

```
:
FTP      mirrorwebHTTP      webcopy
()
```

```
:
( Proxy Throughput)(flag [P])
```

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^hotsheet/(.*)$ http://www.tstimpr
```

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^usa-news\.html$ http://www.quux-
```

```
:
...
```

```
:
```

```
RewriteEngine on
RewriteCond /mirror/of/remotesite/$1
```

-

```
RewriteRule ^http://www\.remotesite\.com/(.*)$ /
```

Intranet

:

```
()Intranet( www2.quux-corp.dom)Internet web  
(www.quux-corp.dom)
```

:

(packet-filtering)

```
ALLOW Host www.quux-corp.dom Port >1024 --> Host w
```

```
DENY Host * Port * --> Host w
```

mod_rewrite

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

:

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

:

“DNS” mod_rewrite:

1. DNS(DNS Round-Robin)

```
BINDDNS www[0-9].foo.comDNSA()
```

```
www0 IN A 1.2.3.1  
www1 IN A 1.2.3.2
```

```

www2    IN    A      1.2.3.3
www3    IN    A      1.2.3.4
www4    IN    A      1.2.3.5
www5    IN    A      1.2.3.6

```

:

```

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

```

```

BIND      www.foo.com BINDwww0-
www6 /DNS www.foo.comwwwN.foo.com
www.foo.com

```

2. DNS

DNS

<http://www.stanford.edu/~schemers/docs/lbnamed/lbnamed.html>
 lbnamedPerl5DNS

3. (Proxy Throughput Round-Robin)

mod_rewriteDNS www0.foo.comwww.foo.com

```

www     IN    CNAME  www0.foo.com.

```

```

www0.foo.comURL5( www1-www5)URL
lb.pl

```

```

RewriteEngine on

```

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

lb.pl

```
#!/path/to/perl
##
## lb.pl -- load balancing script
##

$| = 1;

$name      = "www";      # the hostname base
$first     = 1;         # the first server (not 0)
$last      = 5;         # the last server in the r
$domain    = "foo.dom"; # the domainname

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

##EOF##
```

```
www0.foo.comSSICGlePerl
```

4. /TCP

CiscoLocalDirectorTCP/IP

:

...

:

```
##
##  apache-rproxy.conf -- Apache configuration for
##

#  server type
ServerType          standalone
Listen              8000
MinSpareServers     16
StartServers        16
MaxSpareServers     16
MaxClients          16
MaxRequestsPerChild 100

#  server operation parameters
KeepAlive           on
MaxKeepAliveRequests 100
KeepAliveTimeout    15
Timeout             400
IdentityCheck       off
HostnameLookups     off

#  paths to runtime files
PidFile             /path/to/apache-rproxy.pid
LockFile            /path/to/apache-rproxy.lock
ErrorLog            /path/to/apache-rproxy.elog
CustomLog           /path/to/apache-rproxy.dlog "

#  unused paths
ServerRoot          /tmp
DocumentRoot        /tmp
CacheRoot           /tmp
RewriteLog          /dev/null
TransferLog         /dev/null
```

```

TypesConfig          /dev/null
AccessConfig         /dev/null
ResourceConfig       /dev/null

# speed up and secure processing
<Directory />
Options -FollowSymLinks -SymLinksIfOwnerMatch
AllowOverride None

</Directory>

# the status page for monitoring the reverse proxy
<Location /apache-rproxy-status>
SetHandler server-status
</Location>

# enable the URL rewriting engine
RewriteEngine        on
RewriteLogLevel      0

# define a rewriting map with value-lists where
# mod_rewrite randomly chooses a particular value
RewriteMap            server rnd:/path/to/apache-rproxy.

# make sure the status page is handled locally
# and make sure no one uses our proxy except our
RewriteRule          ^/apache-rproxy-status.* - [L]
RewriteRule          ^(http|ftp)://.* - [F]

# now choose the possible servers for particular
RewriteRule          ^/(.*\.(cgi|html))$ to://${server}
RewriteRule          ^/(.*)$ to://${server}

# and delegate the generated URL by passing it
# through the proxy module
RewriteRule          ^to://([^/]+)/(.*) http://$1/$2

```

```

# and make really sure all other stuff is forbid
# when it should survive the above rules...
RewriteRule      .*                -

# enable the Proxy module without caching
ProxyRequests    on
NoCache          *

# setup URL reverse mapping for redirect reponse
ProxyPassReverse / http://www1.foo.dom/
ProxyPassReverse / http://www2.foo.dom/
ProxyPassReverse / http://www3.foo.dom/
ProxyPassReverse / http://www4.foo.dom/
ProxyPassReverse / http://www5.foo.dom/
ProxyPassReverse / http://www6.foo.dom/

```

```

##
## apache-rproxy.conf-servers -- Apache/mod_rewri
##

# list of backend servers which serve static
# pages (HTML files and Images, etc.)
static      www1.foo.dom|www2.foo.dom|www3.foo.dom|w

# list of backend servers which serve dynamicall
# generated page (CGI programs or mod_perl scrip
dynamic     www5.foo.dom|www6.foo.dom

```

MIME

:

```

CGIApacheMEMECGIURL(          PATH_INFO
QUERY_STRINGS)      .scgi(CGI)  cgiwrapURL
)URL                 /u/user/foo/bar.scgi cg
/~user/foo/bar.scgi/

```

```
RewriteRule ^/[uge]/([^/]+)/\.www/(.+)\.scgi(.*) .  
... /internal/cgi/user/cgiwrap/~$1/$2.scgi$3 [NS,
```

```
wwwlog( access.logURL) wwwidx(URLGlimpse)  
URL /u/user/foo/swwidx
```

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

CGI

:

URLCGI

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

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

```
HREF="*"
```

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

```
" :log"CGI
```

:

```
foo.htmlfoo.cgi /
```

:

```
URLCGI-script CGI-scriptMIME /~quux/foo.html  
/~quux/foo.cgi
```

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ foo.cgi [T=applicatio
```

:

```
()CGI(cronjob)
```

:

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

```
page.htmlnull page.htmlpage.cgi page.cgi
page.html( STDOUT)CGI page.html page.html
(cronjob)
```

:

:

```
! MIMEmwebNPH mod_rewriteURLURLURL
" :refresh"
```

```
RewriteRule ^(/[uqe]/[^/]+/?.*):refresh /intern
```

URL

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

URL

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

NPH-CGI"""

```
#!/sw/bin/perl
##
##  nph-refresh -- NPH/CGI script for auto refresh
##  Copyright (c) 1997 Ralf S. Engelschall, All Ri
##
$| = 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;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C"
    eval "\$$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
    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 no
    exit(0);
}
```

```
sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
    print "Content-type: multipart/x-mixed-replace
&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, $size) = stat($fi
    $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) =
    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##
```

:
Apache<VirtualHost>ISP

:
(Proxy Throughput)(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-forma  
CustomLog /path/to/access_log "%{VHOST}e %h %l %  
:  
  
# enable the rewriting engine in the main server  
RewriteEngine on  
  
# define two maps: one for fixing the URL and on  
# the available virtual hosts with their corresp  
# 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_URL}    !^/commonurl1/.*
RewriteCond    %{REQUEST_URL}    !^/commonurl2/.*
:
RewriteCond    %{REQUEST_URL}    !^/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 1
#       and remember the virtual host for logging p
RewriteRule    ^/(.*)$    %1/$1    [E=VHOST:${lowercas
:

```



Robots

:
robot /robots.txt"robot"robot

:
/~quux/foo/arc/()robotrobotHTTP
User-Agent

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

:
http://www.quux-corp.de/~quux/

:
100%HTTP Referer

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

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

:

:

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

:

Apache

:

Apache web mod_rewrite mod_proxy mod_proxy

...

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

...user@host-dependent:

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

:

(mod_authz_host)

:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^frien
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^frien
```

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^frien
RewriteRule ^/~quux/only-for-friends/ -
```

(Referer)

:

HTTP"Referer"?

:

...

```
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}
```

```
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

... :

```
##
```

```
## deflector.map
```

```
##
```

```
http://www.badguys.com/bad/index.html -
```

```
http://www.badguys.com/bad/index2.html -
```

```
http://www.badguys.com/bad/index3.html http://so
```

(" -")(URL)URL



```
:  
mod_rewriteFOO/BAR/QUUX/
```

```
:  
RewriteMapRewriteMapApache      STDINURL ()  
URL()          STDOUT
```

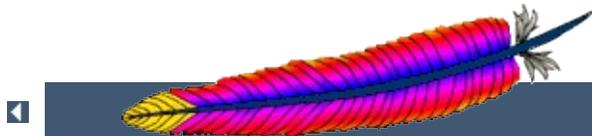
```
RewriteEngine on  
RewriteMap    quux-map      prg:/path/to/map.quux  
RewriteRule   ^/~quux/(.*)$ /~quux/{quux-map:$1}
```

```
#!/path/to/perl
```

```
#  disable buffered I/O which would lead  
#  to deadlocks for the Apache server  
$| = 1;
```

```
#  read URLs one per line from stdin and  
#  generate substitution URL on stdout  
while (<>) {  
    s|^foo/|bar/|;  
    print $_;  
}
```

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



◀ **Apache HTTP Server 2.2**

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

| | | 2006118 |



IPIPIPHHTTPIP

DNSIPApache HTTPIPIPIPI

- "Host"HTTP/1.1HTTP/1.0
- SSLSSL
- IP



<u>core</u>	<u>DocumentRoot</u> <u>NameVirtualHost</u> <u>ServerAlias</u> <u>ServerName</u> <u>ServerPath</u> <u><VirtualHost></u>

```
IP()      NameVirtualHostIP      *" NameVirtualHost
(SSL)"      * :80"      N
IP
```

```
<VirtualHost>  <VirtualHost>NameVirtualHost(IP
" *")  <VirtualHost>  ServerNameDocumentRoot
```

(Mainhost)

```
web  <VirtualHost>  ServerNameDocumentRoot
ServerNameDocumentRoot
```

```
www.domain.tldIP  www.otherdomain.tld
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>
```

```
IP NameVirtualHost<VirtualHost>" *"IPIPIP
```

```
ServerAlias<VirtualHost> <VirtualHost>
ServerAliasweb
```

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

```
domain.tldwww.domain.tld" *"" ?" ServerName
ServerAliasDNSIP
```

```
<VirtualHost> <VirtualHost> (main server)(
<VirtualHost>)
```

```
NameVirtualHostIPIP <VirtualHost>
ServerNameServerAliasIP
```

```
IP NameVirtualHost DocumentRoot
<VirtualHost>
```



IP()

Host

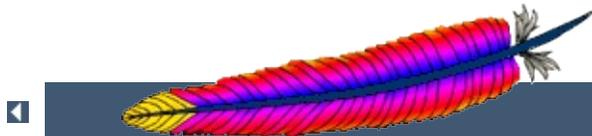
ServerPath

```
NameVirtualHost 111.22.33.44
```

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

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

||||



◀ **Apache HTTP Server 2.2**

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

| | | 2006118 |

IP



" *IP*" **IPIP**("IP""ifconfig")



Apache

apache [httpd](#)

- web [User, Group, Listen, ServerRoot](#)
- IP [Listen](#)("[httpdN-1](#))

- httpd

-



`httpd ListenIP()`

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

IP([DNSApache](#))



[httpd](#) [VirtualHostServerAdmin](#), [ServerName](#),
[DocumentRoot](#), [ErrorLog](#), [TransferLog](#), [CustomLog](#)

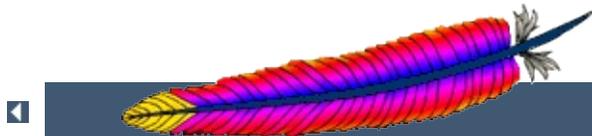
```
<VirtualHost www.smallco.com>
ServerAdmin webmaster@mail.smallco.com
DocumentRoot /groups/smallco/www
ServerName www.smallco.com
ErrorLog /groups/smallco/logs/error_log
TransferLog /groups/smallco/logs/access_log
</VirtualHost>
```

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

IP([DNSApache](#))

```
<VirtualHost> <VirtualHost>
```

[suEXEC](#)[SuexecUserGroup](#)[<VirtualHost>](#)



◀ Apache HTTP Server 2.2

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

| | | 2006118 |



Apache



httpd.conf<VirtualHost>

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

<VirtualHost>

1. Apache
2. DNSApache

()



httpd.conf mod_vhost_alias

```
# "Host:"
UseCanonicalName Off

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

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

UseCanonicalName Off UseCanonicalName DNS IP
IP



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

```
#
VirtualDocumentRoot /www/hosts/%2/docs

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

```
VirtualDocumentRoot mod_vhost_alias
```



Apache <VirtualHost>IP

<VirtualHost>

```
UseCanonicalName Off
```

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

```
<Directory /www/commercial>
```

```
Options FollowSymLinks
```

```
AllowOverride All
```

```
</Directory>
```

```
<Directory /www/homepages>
```

```
Options FollowSymLinks
```

```
AllowOverride None
```

```
</Directory>
```

```
<VirtualHost 111.22.33.44>
```

```
ServerName www.commercial.isp.com
```

```
CustomLog logs/access_log.commercial vcommon
```

```
VirtualDocumentRoot /www/commercial/%0/docs
```

```
VirtualScriptAlias /www/commercial/%0/cgi-bin
```

```
</VirtualHost>
```

```
<VirtualHost 111.22.33.45>
```

```
ServerName www.homepages.isp.com
```

```
CustomLog logs/access_log.homepages vcommon
```

```
VirtualDocumentRoot /www/homepages/%0/docs
```

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

```
</VirtualHost>
```



IPDNSIPApache(ServerName)DNS

```
# IP
UseCanonicalName DNS

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

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



Apache

```
mod_vhost_alias 1.3.6      mod_rewrite"Host:"
```

```
Apache1.3.6"      %V"1.3.0-1.3.3"      %V""      %V"1.3.4  
      UseCanonicalName.htaccess"      %{Host}i"  
" Host:"      :port""      %V"
```



```
httpd.conf      mod_rewrite  mod_rewrite
```

```
    mod_rewriteURI(mod_alias)      mod_rewrite
    ScriptAlias
```

```
# "Host:"
UseCanonicalName Off

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

<Directory /www/hosts>
    # ExecCGI CGIScriptAlias
    Options FollowSymLinks ExecCGI
</Directory>

#

RewriteEngine On

# "Host:"ServerName
RewriteMap lowercase int:tolower

##
# /icons/
RewriteCond %{REQUEST_URI} !^/icons/
# CGI
RewriteCond %{REQUEST_URI} !^/cgi-bin/
# ""
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%
{SERVER_NAME}}/docs/$1

## CGI(MIME)
```

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

```
# ok
```



```
RewriteEngine on

RewriteMap lowercase int:tolower

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

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

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

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

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



mod_rewrite

vhost.map

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

http.conf

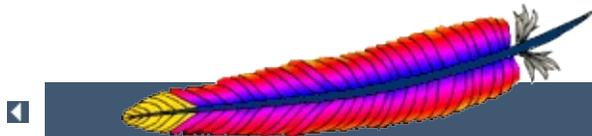
```
RewriteEngine on

RewriteMap lowercase int:tolower

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

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

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



◀ **Apache HTTP Server 2.2**

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

| | | 2006117 |

[IPweb](#)



IPDNS(CNAMES) www.example.comwww.example.org

ApacheDNS DNSIPweb hosts hosts

```
# Apache80
Listen 80

# IP
NameVirtualHost *:80

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

    #

</VirtualHost>

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

    #

</VirtualHost>
```

IP www.example.com ServerName
<VirtualHost>

IP" "*" VirtualHostNameVirtualHost

```
NameVirtualHost 172.20.30.40
```

```
<VirtualHost 172.20.30.40>
```

```
# ...
```

IP"

*"ISPIP"

*"IPIP

IP



IP

IP(172.20.30.40)server.domain.com (172.20.30.50)

```
Listen 80
```

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

```
# IP
NameVirtualHost 172.20.30.50
```

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

```
# ...
```

```
</VirtualHost>
```

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

```
# ...
```

```
</VirtualHost>
```

```
172.20.30.50 172.20.30.50" Host:"
www.example.com
```



```
IP(192.168.1.1172.20.30.40)()
server.example.com(172.20.30.40)( 192.168.1.1)
```

```
<VirtualHost>
```

```
NameVirtualHost 192.168.1.1
NameVirtualHost 172.20.30.40

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

```
<VirtualHost>
```

```
serverserver.example.com
" *"IP
```



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

```
Listen 80
Listen 8080
```

```
NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080
```

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

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

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

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



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

```
Listen 80
```

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

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

```
<VirtualHost>( localhost)
```



IP(172.20.30.40|172.20.30.50)www.example.com
www.example.org 808080

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

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

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

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

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



IP

```
Listen 80
```

```
NameVirtualHost 172.20.30.40
```

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

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

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

```
# IP-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>
```



192.168.111.2

ProxyPreserveHost On

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



" _default_ "

IP/

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

/" _default_/" Host:"(/)

[AliasMatchRewriteRule\(\)](#)

" _default_ "

" _default_"80

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

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

80" _default_"()IP

" _default_ "

80" _default_"

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

80



www.example.org()IP

(172.20.30.50)VirtualHost

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

NameVirtualHost 172.20.30.40

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

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

(IP)()



```
" Host:"HTTP/1.0Apache()URL
```

```
NameVirtualHost 172.20.30.40
```

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

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

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

ServerPath <http://www.sub1.domain.tld/sub1/sub1-vhost>

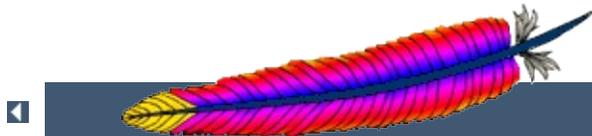
```
" Host:" http://www.sub1.domain.tld/sub1-vhost
```

" Host:"

" Host:" http://www.sub2.domain.tld/sub1/sub1-
vhost

RewriteRule" Host:"URLURL

||||



Apache HTTP Server 2.2

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

| | | 2006117 |



Apache 1.3Apache

[NameVirtualHost1.3](#)



<VirtualHost>(main_server) <VirtualHost>(vhost)

Listen, ServerName, ServerPath, ServerAlias()

Listen80 ServerPathServerAlias ServerNameIP

ListenApacheURI

Apache

VirtualHost Listen" *(DNS A) (address set)

IPNameVirtualHostIP" *"

NameVirtualHostIP NameVirtualHost(CNAME)IP

NameVirtualHostNameVirtualHost"IP:port"

NameVirtualHost

NameVirtualHostVirtualHost IPVirtualHost

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

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

```

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

```

```

<VirtualHost
111.22.33.55>
# server D
...
</VirtualHost>

NameVirtualHost
111.22.33.44
NameVirtualHost
111.22.33.55

```

()

VirtualHost VirtualHostListen

VirtualHostServerAlias(ServerAlias) Listen

IPIP NameVirtualHostIPIP NameVirtualHost
IP

IPIP

1. ServerAdmin, ResourceConfig, AccessConfig, Timeout, KeepAliveTimeout, KeepAlive, MaxKeepAliveRequests, ReceiveBufferSize, SendBufferSize()

2. ()

3.

ServerName [httpdDNS](#) ServerNameIP(*main_server address set*)

ServerName VirtualHost

"_default_" ServerName



IP

(IP) " _default_" _default_"

IP " NameVirtualHost *"

(IP)IP

IP

IP

VirtualHost

(IP) " Host:"

" Host:" ServerNameServerAlias" Host:"
Apache

" Host:"HTTP/1.0 ServerPathURI

IP()

IP TCP/IP(KeepAlive)

URI

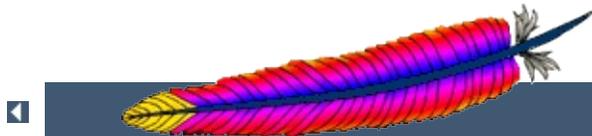
URI URI //URI URI

- `IP` `IP` `NameVirtualHost`
- `IP` `ServerAlias` `ServerPath`
- `IP` `" _default_"` `NameVirtualHost`
- `"` `Host:"Apache`
- `ServerPath` `ServerPath("` `Host:")`
- `IP`
- `IP` `" _default_"` `" _default_"(Listen)(`
`" _default_:*")` `NameVirtualHost *`
- `IP(" _default_")IP(" _default_")`
- `()` `NameVirtualHost` `Host:" " _default_"`
- `VirtualHostDNS` `DNS`
- `ServerName` `DNS`



DNS

- `VirtualHost()`
- `NameVirtualHostVirtualHost`
- `ServerPathsServerPaths""(`
 `"ServerPath/abc/def""ServerPath/abc")`



Apache HTTP Server 2.2

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

| | | 2006117 |

Apache()Apache1020Unix64
(hard-limit)

Apache

1. setrlimit()
2. setrlimit(RLIMIT_NOFILE)(Solaris 2.3)
- 3.
4. stdio256(Solaris 2)

- <VirtualHost>()
- 12Apache

```
#!/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
```

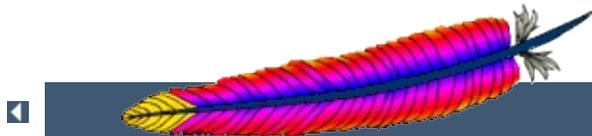
```
( ServerName)( )
```

```
() split-logfileApache support
```

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

```
" .log "
```

||||



Apache HTTP Server 2.2

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

| | | 200612 |

DNSApache

ApacheDNSApacheDNS(())



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

Apache ServerName IP Apache DNS www.abc.dom
DNS (Apache1.2)

www.abc.dom IP 10.0.0.1

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

Apache DNS ServerName (Apache1.2) IP
Apache URL URL

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



Apache2.2DNS
DNS

abc.dom
www.abc.dom1.2Apache

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

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

www.abc.dom10.0.0.1 www.def.dom10.0.0.2 def.domDNS
def.domabc.dom www.def.dom10.0.0.1DNS
www.def.domIP

10.0.0.1(http://www.abc.dom/whateverURL)def.dom
Apache



Apache1.1 Apache [httpd](#) IP [ServerName\(\)](#)C
gethostname("hostname")DNS

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

DNS HOSTRESORDER"local"Apache [mod_env](#)CGIman
FAQ



-
- VirtualHostIP
 - ListenIP
 - ServerName
 - `<VirtualHost _default_*>`



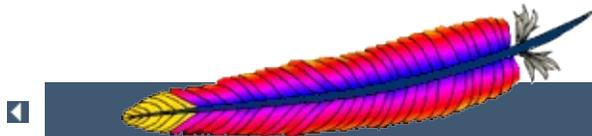
DNSApache1.2DNSInternetIP

DNSDNS(FTP TCP""DNS)

IPDNS

HTTP/1.1HostIPwebDNS19973web

|||



Apache HTTP Server 2.2

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

| | | 2006116 |

SSL/TLS

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

WebHTTPApacheSSL

[mod_ssl](#)

[Introducing SSL and Certificates using SSL](#) Frederick J. Hirsch

Open Group Research Institute 1997

[Web Security](#)

[Matter of Trust](#), World Wide Web Journal, Volume 2, Issue 3,

Summer 1997

[Frederick Hirsch](#)()

[Ralf S.](#)

[Engelschall](#)([mod_ssl](#))



SSL()([AC96](#))

Alice

Alice

()

Alice()

AliceAlice

Alice

AliceAlice

()

()Alice



AliceAlice

(Certificate Authority)

1([Distinguished Name])

1: Certificate Information

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

X.509[[X509](#)]([2](#))

2: Distinguished Name Information

DN Field	Abbrev.	Description	Example
Common Name	CN	Name being certified	CN=Joe Average
Organization or Company	O	Name is associated with this organization	O=Snake Oil, Ltd.
Organizational Unit	OU	Name is associated with this organization unit, such as a department	OU=Research Institute
City/Locality	L	Name is located in this City	L=Snake City

State/Province	ST	Name is located in this State/Province	ST=Desert
Country	C	Name is located in this Country (ISO code)	C=XZ

NetscapeCommon Name

*.snakeoil.com

ASN.1[X208] [PKCS](Basic Encoding Rules[BER])
(Distinguished Encoding Rules[DER])Base64[
PEM("Privacy Enhanced Mail")

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

```
-----BEGIN CERTIFICATE-----
MIIC7jCCAlegAwIBAgIBATANBgkqhkiG9w0BAQQFADCBqTELMAkGA:
FTATBgNVBAGTDFNuYWwtlIERlc2VydeETMBEGA1UEBxMKU25ha2UgV(
A1UEChMOU25ha2UgT2lsLCBMdGQxHjAcBgNVBAsTFUNlcnRpZm1jY(
cm10eTEVMBMGA1UEAxMMU25ha2UgT2lsIENBMR4wHAYJKoZIhvcNA(
bmFrZW9pbC5kb20wHhcNOTGxMDIxMDg10DM2WhcNOTkxMDIxMDg10I
MAkGA1UEBhMCWFkxFTATBgNVBAGTDFNuYWwtlIERlc2VydeETMBEGA:
a2UgVG93bjEXMBUGA1UEChMOU25ha2UgT2lsLCBMdGQxHjAcBgNVB(
cnZlcjBUZWFtMRkwFwYDVQQDExB3d3cuc25ha2VvaWwuZG9tMR8wH(
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA(
gQDH9Ge/s2zcH+da+rPTx/DPRp3xGjHZ4GG6pCmvADIEtBtKBFAcZ(
vKR+yy5DGQiijsh1D/j8HlGE+q4TZ80Fk7BNBFazHxFbYI40KMicx(
lwoANFlAzlSdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABO(
HRMECDAGAQH/AgEAMBEGCWCGSAGG+EIBAQQEAwIAQDANBgkqhkiG9w(
gQAZUIHAL4D09oE6Lv2k56Gp380BDuILvwLg1v1KL8mQR+KFjghCr(
2q2QoyulCgSzHbEGmi0EsdKpfg6mp0penssIFePYNI+/8u9HT4LuK(
dUHziCxBVC1lnHyYGjDuAMhe396lYAn8bClD1/L4NMGBCQ==
-----END CERTIFICATE-----
```

AliceAlice

Alice""

CA

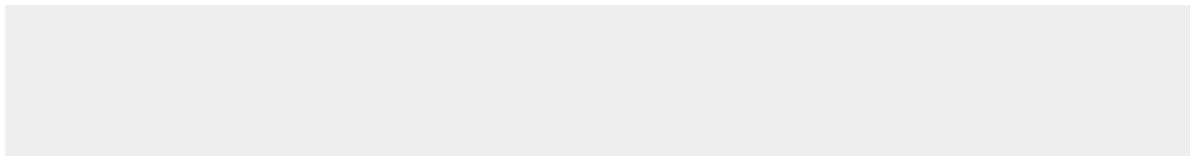
""--

[ThawteVeriSign](#)

-
-
-

InternetIntranet

([Certificate Revocation ListsCRL])Alice
Alice()



(TCP/IP)(HTTP)SSL

4: Versions of the SSL protocol

Version	Source	Description	Browser Support
SSL v2.0	Vendor Standard (from Netscape Corp.) [SSL2]	First SSL protocol for which implementations exists	- NS Navigator 1.x/2.x - MS IE 3.x - Lynx/2.8+OpenSSL
SSL v3.0	Expired Internet Draft (from Netscape Corp.) [SSL3]	Revisions to prevent specific security attacks, add non-RSA ciphers, and support for certificate chains	- NS Navigator 2.x/3.x/4.x - MS IE 3.x/4.x - Lynx/2.8+OpenSSL
TLS v1.0	Proposed Internet Standard (from IETF) [TLS1]	Revision of SSL 3.0 to update the MAC layer to HMAC, add block padding for block ciphers, message order standardization and more alert messages.	- Lynx/2.8+OpenSSL

[4SSLSSL3.0SSL3.0Internet Engineering Task Force\(IETF\)\[](#) [TLS\]](#)

SSL [Figure 1SSL](#)

SSLSSL()

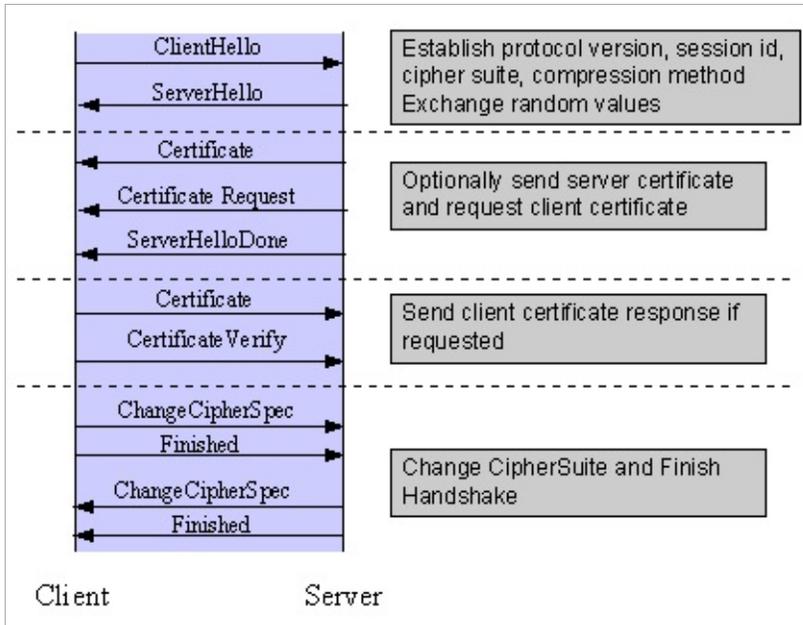


Figure 1: Simplified SSL Handshake Sequence

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

SSL3.031

-
-
- (Message Authentication Code[MAC])

SSL2.0RSASSL3.0RSA-Diffie-Hellman

() [[AC96](#), p516]

SSL()

- No encryption
- Stream Ciphers
 - RC4 with 40-bit keys
 - RC4 with 128-bit keys
- CBC Block Ciphers
 - RC2 with 40 bit key
 - DES with 40 bit key
 - DES with 56 bit key
 - Triple-DES with 168 bit key
 - Idea (128 bit key)
 - Fortezza (96 bit key)

"CBC" Cipher Block Chaining "DES" Data Encryption
Standard [[AC96](#), ch12] (DES403DES_EDE) "Idea" "RC2"
RSADSI [[AC96](#), ch13]

SSL

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

(MAC)

- *SSL Handshake Protocol*
- *SSL Change Cipher Spec Protocol*
- *SSL Alert Protocol*

SSL Record Protocol [Figure 2](#)

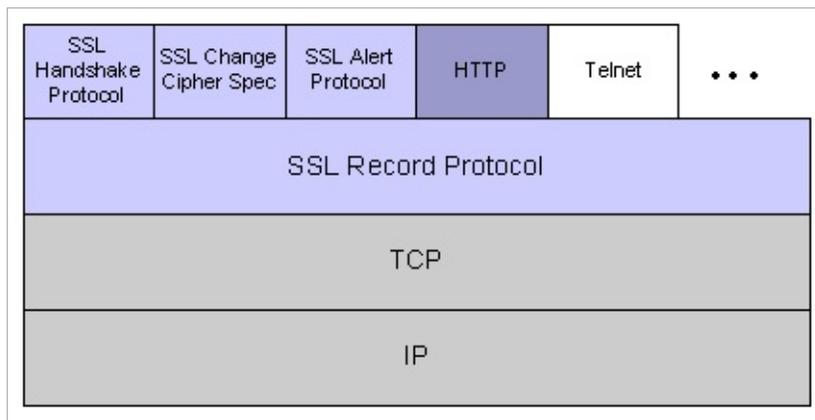


Figure 2: SSL Protocol Stack

SSLNull

SSL [Figure 3](#)SSL(SSL)

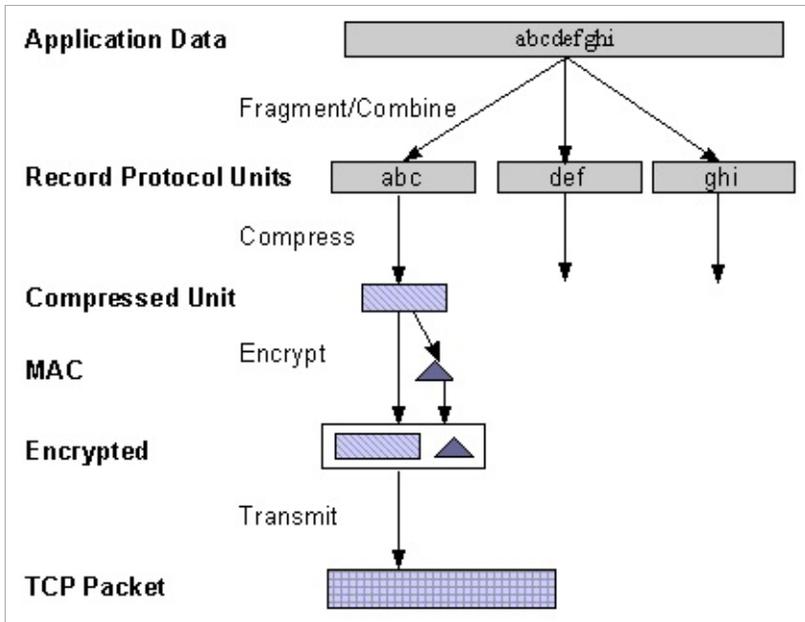


Figure 3: SSL Record Protocol

HTTP

SSLHTTPHTTPHTTPSSL(HTTPS)URL
 (443) [mod_ssl](#)Apache...

httpshttp



[AC96]

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

[X208]

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

[X509]

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

[PKCS]

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

[MIME]

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

[SSL2]

Kipp E.B. Hickman, *"The SSL Protocol"*, 1995. See http://www.netscape.com/eng/security/SSL_2.html.

[SSL3]

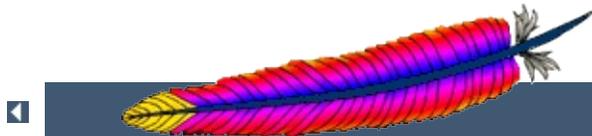
Alan O. Freier, Philip Karlton, Paul C. Kocher, *"The SSL Protocol Version 3.0"*, 1996. See <http://www.netscape.com/eng/ssl3/draft302.txt>.

[TLS1]

Tim Dierks, Christopher Allen, *"The TLS Protocol Version 1.0"*,

1999. See <http://ietf.org/rfc/rfc2246.txt>.

| | | |



Apache HTTP Server 2.2

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

| | | 2006116 |

SSL/TLS

PC

--

SSLmod_sslApacheSSLBen Laurie

mod_ssl)RedHat

[Secure Web Server](#)(mod_ssl)Covale

[Raven SSL Module](#)(mod_ssl)C2Net [Stronghold](#)(Stringhold2.x

SiouxStronghold3.xmod_ssl)

mod_sslmod_ssl



SSL [1](#)Apache-SSL1.xmod_ssl2.0.xSioux1.x
 Stronghold2.xmod_ssl

1:

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

SSL_X509VerifyDir <i>arg</i>	<i>arg</i>	
SSL_Log <i>file</i>	SSLLogFile <i>file</i>	
SSL_Connect <i>flag</i>	SSLEngine <i>flag</i>	
SSL_ClientAuth <i>arg</i>	SSLVerifyClient <i>arg</i>	
SSL_X509VerifyDepth <i>arg</i>	SSLVerifyDepth <i>arg</i>	
SSL_FetchKeyPhraseFrom <i>arg</i>	-	SSLP
SSL_SessionDir <i>dir</i>	-	SSLS
SSL_Require <i>expr</i>	-	SSLR
SSL_CertFileType <i>arg</i>	-	
SSL_KeyFileType <i>arg</i>	-	
SSL_X509VerifyPolicy <i>arg</i>	-	
SSL_LogX509Attributes <i>arg</i>	-	
Stronghold 2.x :		
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>	-	



" SSLOptions +CompatEnvVars"mod_ssl

2

2:

	mod_ssl
SSL_PROTOCOL_VERSION	SSL_PROTOCOL
SSLEAY_VERSION	SSL_VERSION_LIBRARY
HTTPS_SECRETKEYSIZE	SSL_CIPHER_USEKEYSIZE
HTTPS_KEYSIZE	SSL_CIPHER_ALGKEYSIZE
HTTPS_CIPHER	SSL_CIPHER
HTTPS_EXPORT	SSL_CIPHER_EXPORT
SSL_SERVER_KEY_SIZE	SSL_CIPHER_ALGKEYSIZE
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_SERIAL
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_Email
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_Email
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_SERIAL
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_Email
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_Email
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_ALGKEYSIZE
SSL_SECKEYSIZE	SSL_CIPHER_USEKEYSIZE
SSL_SSLEAY_VERSION	SSL_VERSION_LIBRARY

SSL_STRONG_CRYPT0	-	m
SSL_SERVER_KEY_EXP	-	m
SSL_SERVER_KEY_ALGORITHM	-	m
SSL_SERVER_KEY_SIZE	-	m
SSL_SERVER_SESSIONDIR	-	m
SSL_SERVER_CERTIFICATELOGDIR	-	m
SSL_SERVER_CERTFILE	-	m
SSL_SERVER_KEYFILE	-	m
SSL_SERVER_KEYFILETYPE	-	m
SSL_CLIENT_KEY_EXP	-	m
SSL_CLIENT_KEY_ALGORITHM	-	m
SSL_CLIENT_KEY_SIZE	-	m

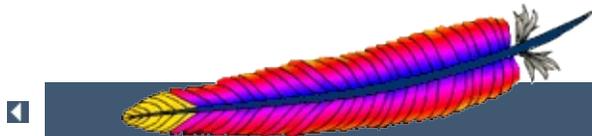


mod_sslApache(DSO)

" *{name}*c" [3](#)

3:

Function Call	
%...{version}c	SSL
%...{cipher}c	SSL
%...{subjectdn}c	Subject Distinguished Name
%...{issuerdn}c	Issuer Distinguished Name
%...{errcode}c	()
%...{errstr}c	()



Apache HTTP Server 2.2

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

| | | 2006116 |

SSL/TLS...?

--

SSLHTTPApacheSSLweb



-
- [SSLv2](#)
 -
 -
 -

SSLv2

SSLv2

httpd.conf

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

SSL

httpd.conf

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

SSL

(Server Gated Cryptography[SGC])mod_ssl README.GlobalID
VerisignCAIDHTTP

httpd.conf

```
# SGC
SSLCipherSuite
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

<Directory /usr/local/apache2/htdocs>
```

```
#
SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
</Directory>
```

SSLURL

[SSLCipherSuite](#)mod_sslSSL

```
#
SSLCipherSuite
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

<Location /strong/area>
# https://hostname/strong/area/
SSLCipherSuite HIGH:MEDIUM
</Location>
```



-
-
-
- [intranetinternet](#)

()IntranetCA

ca.crt

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
```

URL

[mod_ssl](#)

httpd.conf

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

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

URL

Distinguished Name (DN)

[mod_auth_basic](#)[SSLRequire](#)
DN

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
AuthBasicProvider     file
AuthUserFile          /usr/local/apache2/conf/httpd.passwd
require               valid-user
</Directory>
```

httpd.passwd

```
/C=DE/L=Munich/O=Snake Oil, Ltd./OU=Staff/CN=Foo:xxj3:
/C=US/L=S.F./O=Snake Oil, Ltd./OU=CA/CN=Bar:xxj31ZMTZ:
/C=US/L=L.A./O=Snake Oil, Ltd./OU=Dev/CN=Quux:xxj31ZMTZ:
```

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"
                        and %{SSL_CLIENT_S_DN_OU} in {"Staff",
</Directory>

```

InternetHTTPSIntranetIntranetHTTP

IntranetIP192.160.1.0/24IntranetURL /subarea HTTPS(HTTPS
HTTP)

httpd.conf

```

SSLCACertificateFile conf/ssl.crt/company-ca.crt

<Directory /usr/local/apache2/htdocs>
# subareaIntranet
Order                deny,allow
Deny                from all
Allow                from 192.168.1.0/24
</Directory>

<Directory /usr/local/apache2/htdocs/subarea>
# subareaIntranet
# InternetHTTPS+Strong-Cipher+Password
# HTTPS+Strong-Cipher+Client-Certificate

# HTTPS
#
SSLVerifyClient        optional
SSLVerifyDepth        1
SSLOptions            +FakeBasicAuth +StrictRequire
SSLRequire            %{SSL_CIPHER_USEKEYSIZE} >= 128

# InternetHTTPS
RewriteEngine        on
RewriteCond            %{REMOTE_ADDR} !^192\.168\.1\.[0

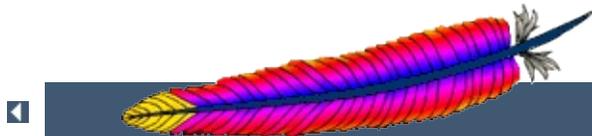
```

```
RewriteCond      %{HTTPS} !=on
RewriteRule      .* - [F]

#
Satisfy          any

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

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



Apache HTTP Server 2.2

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

| | < > | ??? |

SSL/TLS Strong Encryption: FAQ

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

-- Claude Levi-Strauss

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

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



- [What is the history of mod_ssl?](#)
- [mod_ssl and Year 2000?](#)
- [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/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](#) directives.



-
- [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 over one instance of Apache - one responding to requests on port 80 and speaking HTTP and the other responding to requests on port 443 speaking HTTPS.

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 like this (for port 666):
`https://secure.server.dom:666/`

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 for 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 on ports 80 and 443 as follows:

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

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

Because you connected with HTTP to the HTTPS port, i.e. you used an URL of the form "http://" instead of "https://". This also happens the other way round when you connect via HTTPS to a HTTP port, i.e. when you try to use "https://" on a server that doesn't support SSL (on this port). Make sure you are connecting to a virtual server that supports SSL, which is probably the IP associated with your hostname, not localhost (127.0.0.1).

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

This can happen for various reasons. The most common mistakes include starting Apache with just `apachectl start` (or `httpd`) instead of `apachectl startssl` (or `httpd -DSSL`). Your configuration may also be incorrect. Please make sure that your `Listen` directives match your `<VirtualHost>` directives. If all else fails, please start afresh, using the default configuration provided by `mod_ssl`.

Why are the SSL_XXX variables not available to my CGI & SSI scripts?

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

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

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

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

This rewrite ruleset lets you use hyperlinks of the form ``, to switch to HTTPS in a relative link.



- [What are RSA Private Keys, CSRs and Certificates?](#)
- [Is there a difference on startup between the original 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 getcagetverisign 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 [chapter](#) for a general description of the SSL protocol.

Is there a difference on startup between the original 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.

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 server . crt files:
\$ openssl req -new -x509 -nodes -out server.crt -keyout server.key

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

```
SSLCertificateFile    /path/to/this/se
SSLCertificateKeyFile /path/to/this/se
```

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.keyserver.crt`.

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

```
SSLCertificateFile    /path/to/this/server.c  
SSLCertificateKeyFile /path/to/this/server.k
```

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.shCA.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 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 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 `getcagetverisign` 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:-
```

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/2.2/ssl/ssl_faq.html

First check the F.A.Q. ([this text](#)). If your problem is a common one, it may have been answered several times before, and been included in this doc.

Postings from the modssl-users Support Mailing List

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

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

What support contacts are available in case of mod_ssl problems?

The following lists all support possibilities for mod_ssl, in order of preference. Please go through these possibilities *in this order* - don't just pick the one you like the look of.

1. *Send a Problem Report to the modssl-users Support Mailing List*

modssl-users@modssl.org

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

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

users@httpd.apache.org

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

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

http://httpd.apache.org/bug_report.html

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

What information should I provide when writing a bug report?

You should always provide at least the following information:

Apache and OpenSSL version information

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

The details on how you built and installed Apache+mod_ssl+OpenSSL

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

In case of core dumps please include a Backtrace

If your Apache+mod_ssl+OpenSSL dumps its core, please attach a stack-frame "backtrace" (see [below](#) for information on how to get this). Without this information, the reason for your core dump cannot be found

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?

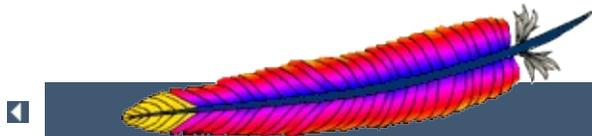
In general no, at least not unless you provide more details about the code location where Apache dumped core. What is usually always required in order to help you is a backtrace (see next question). Without this information it is mostly impossible to find the problem and help you in fixing it.

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/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.



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [.../](#)

| | | 200618 |

(Authentication)(Authorization)



-
- ([AuthType](#))
 - [mod_auth_basic](#)
 - [mod_auth_digest](#)
 - - [mod_authn_alias](#)
 - [mod_authn_anon](#)
 - [mod_authn_dbd](#)
 - [mod_authn_dbm](#)
 - [mod_authn_default](#)
 - [mod_authn_file](#)
 - [mod_authnz_ldap](#)
 - ([Require](#))
 - [mod_authnz_ldap](#)
 - [mod_authz_dbm](#)
 - [mod_authz_default](#)
 - [mod_authz_groupfile](#)
 - [mod_authz_owner](#)
 - [mod_authz_user](#)

[mod_authnz_ldap](#) [mod_authn_alias](#)

[mod_authz_hostIP](#)





||||



(<Directory>)(.htaccess)

.htaccess AllowOverride

AllowOverride

AllowOverride AuthConfig



```
    /usr/local/apache/htdocs  
/usr/local/apache/passwd
```

Apachebin[htpasswd](#)

```
htpasswd -c /usr/local/apache/passwd/passwords  
rbowen
```

[htpasswd](#)

```
# htpasswd -c /usr/local/apache/passwd/passwords  
rbowen  
New password: mypassword  
Re-type new password: mypassword  
Adding password for user rbowen
```

[htpasswd](#) /usr/local/apache/bin/htpasswd

```
httpd.conf.htaccess  
/usr/local/apache/htdocs/secret  
/usr/local/apache/htdocs/secret/.htaccesshttpd.conf  
<Directory /usr/local/apache/apache/htdocs/secret>
```

```
AuthType Basic  
AuthName "Restricted Files"  
AuthUserFile /usr/local/apache/passwd/passwords  
Require user rbowen
```

```
AuthType mod_auth_basicBasic Basic  
Apache" AuthType Digest" mod_auth_diges
```

```
AuthName(Realm)
```

"Restricted Files" "Restricted Files"

AuthUserFile htpasswdApache
mod_authn_dbmAuthDBMUserFile dbmmanage Apache

Require Require



```
( rbowen)      AuthGroupFile
```

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

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

```
(          -c)
```

```
.htaccess
```

```
AuthType Basic  
AuthName "By Invitation Only"  
AuthUserFile /usr/local/apache/passwd/passwords  
AuthGroupFile /usr/local/apache/passwd/groups  
Require group GroupName
```

```
GroupNamepassword
```

```
Require valid-user
```

```
Require user rbowen Apache()  
AuthUserFile
```



Basic



AllowDeny OrderApache

```
Allow from address
```

*address*IP(IP)()IP

```
Deny from 205.252.46.165
```

IP

```
Deny from host.example.com
```

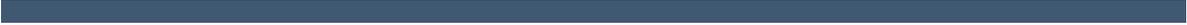
```
Deny from 192.101.205  
Deny from cyberthugs.com moreidiots.com  
Deny from ke
```

OrderDenyAllow

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

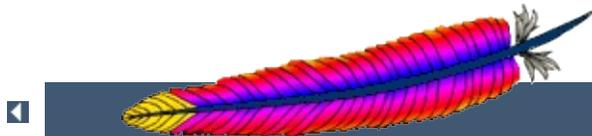
Allow





mod_auth_basic mod_authz_host mod_authn_alias

| | | |



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [.../](#)

| | | 200618 |

CGI



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

CGI()webCGICGIApache webCGICGI



CGIApacheCGI

ScriptAlias

ScriptAliasApacheCGIApacheCGI

ScriptAlias

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

```
Apache httpd.conf ScriptAliasAliasURL  
DocumentRoot ScriptAliasURLCGIApache /cgi-  
bin//usr/local/apache2/cgi-bin/CGI
```

```
URL http://www.example.com/cgi-bin/test.pl  
Apache /usr/local/apache2/cgi-bin/test.pl  
Apache
```

ScriptAliasCGI

```
CGI ScriptAliasCGICGI UserDir  
CGI cgi-binCGI
```

```
CGI AddHandlerSetHandlercgi-script Options  
ExecCGI
```

OptionsCGI

OptionsCGI

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

```
ApacheCGICGI AddHandlercgiplCGI
```

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

.htaccess

[.htaccess](#)httpd.confCGI

" .cgi"CGI

```
<Directory /home/*/public_html>  
  Options +ExecCGI  
  AddHandler cgi-script .cgi  
</Directory>
```

cgi-binCGI

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



CGI""

CGIHTTP [MIME](#)

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

HTMLHTMLgifHTML

CGI

CGI

CGI first.pl cgi-bin

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

PerlApache /usr/bin/perl(shell)

HTTP"Hello, World."

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

Hello, World.



CGI

CGI

CGI Content-Type

CGI "POST Method Not Allowed"

ApacheCGI [Apache](#)

"Forbidden"

[Apache](#)

"Internal Server Error"

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

(nobodywww) nobody

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

shell PATH shell

CGIweb PATH CGI(sendmail)shellCGI

CGI(perl)

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

CGI Apache

CGI

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

(perlshellApache)

HTTP Content-TypeApache Premature end of
script headers [CGI](#)

Suexec

[suexec](#)CGI [suexec](#)CGI Premature end of script
headers

suexec apachectl -V SUEXEC_BINApache [suexec](#)
suexec

suexec() SUEXEC_BIN[suexec](#) [suexec](#) suexec
-V suexec



CGI()"Hello, World"

() env

CGI(Netscape|ELynx)(Apache|ISWebSite)CGI

CGI- <http://hoo.hoo.ncsa.uiuc.edu/cgi/env.html>

CGIApache cgi-binApache

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

STDIN STDOUT

(STDIN)(STDOUT) STDIN STDOUT

POSTCGI STDINCGI

""(=)(&)"&""="

name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Sql

URL QUERY_STRING GETHTML FORMMETHOD GETPOST

CGI



CGI

PerlCGI [CPAN](#) CGI.pm CGI::Lite

CCGI CGIC <http://www.boutell.com/cgic/>



CGIUsenet comp.infosystems.www.authoring.cgiCGIHTML

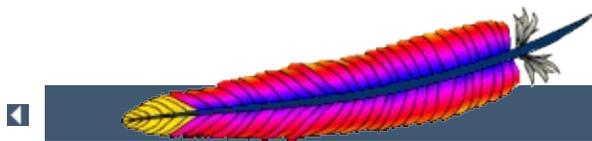
Writers Guild <http://www.hwg.org/li>

CGICGI [NCSA Common Gateway Interface RFC project](http://www.nps.gov/learn/visit/visit.cgi)

CGICGI

CGIApache bugApache

| | | |



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [.../](#)

| | | 200619 |

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)SSI HTMLSSI

SSISSI



SSIHHTMLHTMLCGI

SSI SSI



SSI httpd.conf.htaccess

Options +Includes

SSI [Options](#)SSI Options

SSIApacheApache .shtml

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

.shtmlSSI

[XBitHack](#)

XBitHack on

[XBitHack](#)ApacheSSI chmodSSI

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

.shtmlApache .htmlSSI [XBitHack](#) Apache
SSI

Windows

ApacheSSIHTTP

1. XBitHack Full Apache
2. [mod expires](#)



SSI

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

HTMLSSI HTMLSSI

SSI

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

echoCGI set

 configtimefmt

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

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

timefmt

CGI

SSICGI""

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



HTMLSSI

?

SSIHTMLSSI

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

```
ssi.shtml          LAST_MODIFIED
```

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

timefmt [googlestrftime](#)

```
/          include includefilevirtual file("/")
).."          virtualURL "
```

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

SSI LAST_MODIFIEDSSI include



config

SSI

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

configerrmsg

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

configsizefmt bytesKbMb (abbrev)



CGISSI execSSIshell(/bin/sh Win32DOS shell)

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

Windows

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

Windows dir"< dir>"

exec"" OptionsIncludesNOEXEC exec
SSI



Apache SSI

Apache1.2Apache1.2

set

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

```
( LAST_MODIFIED)"$"
```

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

```
"$"\$"
```

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

```
()
```

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

SSI [mod_include](#)if, elif, else, endif

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

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

```
test_condition"())
```

```
mod_include
```

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

```
MacintoshInternet Explorer"Mac""InternetExplorer"
```

SSI

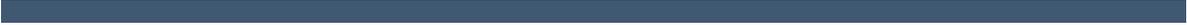
```
<!--#if expr="${Mac} && ${InternetExplorer}" -->  
Apologetic text goes here  
<!--#else -->  
Cool JavaScript code goes here  
<!--#endif -->
```

```
MacIEMacIEJavaScript
```

```
()Apache
```

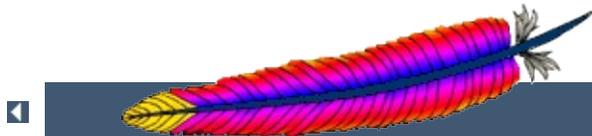
```
SetEnvIfCGI
```





SSICGI





Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [.../](#)

| | | 200618 |

.htaccess

.htaccess



<u>core</u>	<u>AccessFileName</u>
<u>mod_authn_file</u>	<u>AllowOverride</u>
<u>mod_authz_groupfile</u>	<u>Options</u>
<u>mod_cgi</u>	<u>AddHandler</u>
<u>mod_include</u>	<u>SetHandler</u>
<u>mod_mime</u>	<u>AuthType</u>
	<u>AuthName</u>
	<u>AuthUserFile</u>
	<u>AuthGroupFile</u>
	<u>Require</u>



.htaccess("")

```
.htaccess AccessFileName .config
```

```
AccessFileName .config
```

```
.htaccess AllowOverride.htaccess .htaccess  
AllowOverride
```

```
AddDefaultCharset.htaccess("") FileInfo  
.htaccess AllowOverride FileInfo
```

```
server config, virtual host, directory, .htaccess
```

```
FileInfo
```

```
.htaccess""".htaccess"
```



```
.htaccess .htaccess
.htaccessroot .htaccessISP
.htaccess .htaccess <Directory>
.htaccess
AllowOverride .htaccessApache .htaccess
.htaccess .htaccess
Apache .htaccess( ) /www/htdocs/example
Apache
```

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

```
4(" /" .htaccess )
```

AllowOverride

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

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

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

httpd.conf

```
<Directory /www/htdocs/example>
```

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

Apache

AllowOverridenone.htaccess

```
AllowOverride None
```



```
.htaccess.htaccess      .htaccess      .htaccess
.htaccess
```

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

```
Options +ExecCGI
```

```
("    AllowOverride Options" .htaccess" Options)
```

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

```
Options Includes
```

```
.htaccess /www/htdocs/example1/example2CGI
Options Includes
```

.htaccess

```
() .htaccess<Directory>      AllowOverride
.htaccess
```

```
<Directory />
  Allowoverride All
</Directory>
```

```
<Location />
  Options +IncludesNoExec -ExecCGI
</Location>
```



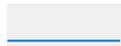
.htaccess <Directory> .htaccess
.htaccess

.htaccess

.htaccess

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

AllowOverride AuthConfig



.htaccess(SSl) .htaccess

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

```
AllowOverride Options AllowOverride FileInfo
```

[SSl](#)



.htaccessCGI

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

CGI

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

```
AllowOverride Options AllowOverride FileInfo
```

CGI [CGI](#)

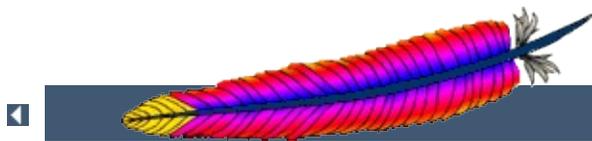


.htaccess

AllowOverride AllowOverride None .htaccess
AllowOverride None

Apache .htaccess

||||



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [.../](#)

| | | 200619 |

UserDirURL http://example.com/~username/
" username" UserDir



<u>mod userdir</u>	<u>UserDir</u> <u>DirectoryMatch</u> <u>AllowOverride</u>



UserDir

```
UserDir public_html
```

```
URL http://example.com/~rbowen/file.html  
    /home/rbowen/public_html/file.html
```

```
UserDir /var/html
```

```
URL http://example.com/~rbowen/file.html  
    /var/html/rbowen/file.html
```

(*)

```
UserDir /var/www/*/docs
```

```
URL http://example.com/~rbowen/file.html  
    /var/www/rbowen/docs/file.html
```



UserDir

```
UserDir enabled  
UserDir disabled root jro fish
```

disabled

```
UserDir disabled  
UserDir enabled rbowen krietz
```

UserDir



<Directory>"cgi" cgi-bin

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

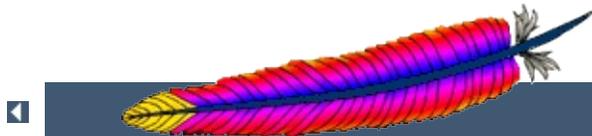
"" UserDirpublic_html CGIexample.cgiURL

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



`.htaccess` `AllowOverride` `.htaccess`

||||



Apache HTTP Server 2.2

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

| | | 2006112 |

Microsoft WindowsApache

Microsoft WindowsApache 2.0 [bug](#)

ApacheWindows Apache(bugs) [WindowsApache](#)

Windows

- **Windows NT:** NTMicrosoft WindowsWindows NT, Windows 2000, Windows XP, Windows.NET Server 2003
- **Windows 9x:** Microsoft Windows Windows 95 , Windows 98, Windows ME



Apache 2.0 Windows NTx86 IntelAMD Apache Windows 9x

TCP/IP Windows 95 "Winsock2" "Winsock2" for Windows
95

NT 4.0 Service Pack 6 Service Pack 4 TCP/IP Winsock Service
Pack



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

.msi Apache for Windows Microsoft Installer Apache

.zip Microsoft Visual C++ (Visual Studio)



ApacheMicrosoft Installer 1.2 Windows 9x
2.0 Windows NT 4.020002.0

Microsoft Installer
Windows XP/2003

Apache 2.0 1.3

2.0Apache 2.0

[Apache](#)

Apache .msi

1. **Network Domain** DNSDNS server.mydomain.net
mydomain.net
2. **Server Name** DNS server.mydomain.net
3. **Administrator's Email Address** email
4. **For whom to install Apache** Apache80(Apache)
" for All Users, on Port 80, as a Service -
Recommended"Apache80WWW" only for the
Current User, on Port 8080, when started
Manually"
5. **The installation type** Typical Custom13MB
6. **Where to install Apache** C:\Program Files\Apache
GroupApache2

Apache conf .default
conf\httpd.conf conf\httpd.conf
conf\httpd.conf.default .default

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

Apache confApache htdocs



UnixApache confWindows

Apache for Windows

- Apache for WindowsUnix Apache

MaxRequestsPerChildUnix Unix
MaxRequestsPerChild 0

httpd.conf

ThreadsPerChild ThreadsPerChild 50

- WindowsUnix ApacheUnixApache

- Apache for WindowsApach \Apache2\modules
LoadModule (access.conf)

LoadModule status_module modules/mod_status.so

- ApacheISAPI(Internet Server Applications Programming Interface)Microsoft IISWindows
- CGIApache ScriptInterpreterSource
- Windows.htaccess AccessFilename
- Windows NTApacheWindows(event log)Apache error.log
""MMCWindows

Windows 9x



ApacheWindows NT

Apache"for all users"Apache"only for the Current User"

ApacheAdministrators

Apache Service MonitorApacheApacheApache

ApachebinApacheWindows NT

```
apache -k install
```

Apache

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

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

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

Apache

```
apache -k uninstall
```

Apache

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

ApacheApache Service Monitor NET START

Apache2 NET STOP Apache2 WindowsApache

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

ApacheApache

```
apache -k start
```

Apache

```
apache -k stop
```

```
apache -k shutdown
```

Apache

```
apache -k restart
```

Apache(LocalSystem) LocalSystemWindows
DCOMsecure RPC

LocalSystemApacheApache

ApacheApache

- 1.
2. Windows 2000/XP/2003""""MMC
3. Users
4. (RX)(htdocscgi-bin)
5. ApacheLogs//(RWD)
6. Apache.exe(RX)

```
Apache(RX)Apache2 logs//(RWD)
```

webApache Apache

```
2186'''
```

ApacheWindows Apache

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

Apache [Apache](#)

ApacheWindows 9xWindows NT Apache

'''

Apache

```
Apache -n "" -k start
```

Apache httpd.conf

Windows 9xNET STARTNET STOPApache

ApacheWindows 9xApacheWindows 9xApacheWindows 9x
httpdwebApacheintranet



ApacheWindows 9xApache

Apache

```
apache
```

ApacheCtl+C

```
--> --> Apache HTTP Server 2.2.xx -->  
Control Apache Server Apache Apache Apache  
Ctl+CApacheApache
```

Apache

```
apache -k shutdown
```

Ctl+CApache

ApacheApache

```
apache -k restart
```

```
UnixApacheUnix    kill -TERM pid  kill -USR1 pid  
-k Unix    kill
```

ApacheApachebin apache error.logApache

```
c:  
cd "\Program Files\Apache Group\Apache2\bin"  
apache
```

ApacheCtl+C

```
cd ..\logs
more < error.log
```

Apache

- -f

```
apache -f "c:\my server
files\anotherconfig.conf"
```

```
apache -f files\anotherconfig.conf
```

- -n Apache

```
apache -n ""
```

ServerRoot

```
-f -n Apache conf\httpd.conf -V
Apache SERVER_CONFIG_FILE
```

```
apache -V
```

ApacheServerRoot

1. -C ServerRoot
2. -d
- 3.
- 4.
5. /apache apache -V HTTPD_ROOT

"for all users"

HKEY_LOCAL_MACHINE

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache
Group\Apache\2.0.43
```

"for the current user only"

HKEY_CURRENT_USER

```
HKEY_CURRENT_USER\SOFTWARE\Apache
Group\Apache\2.0.43
```

Apache

confServerRootApache httpd.conf ServerRoot
ApacheApache



Apache() 80(Listen URL

http://localhost/

Apache logs error.logDNSURL

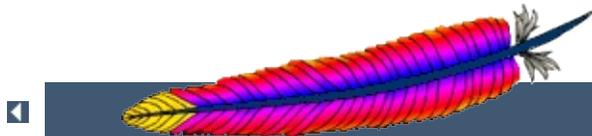
http://127.0.0.1/

Apache80(8080)URL

http://127.0.0.1:8080/

confApache NTApacheApache

ApacheTCP/IP()webBlackIceApache ApacheTCP/IP



Apache HTTP Server 2.2

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

| | | 2006112 |

Microsoft WindowsApache

Apache [Microsoft WindowsApache](#)



Apache

-

50MBApache10MB

- Microsoft Visual C++ 5.0

Visual StudioApache PATH, INCLUDE, LIB
vcvars32

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

- Windows Platform SDK

Visual C++ 5.0 Microsoft Windows Platform SDKApache
setenv

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

Visual C++ 6.0 Platform SDK

```
Windows Platform SDKApache      mod_isapiSDK  
MSVC++ 5.0 Apache                mod_isapi  
http://msdn.microsoft.com/downloads/sdks/platform/platform.asp  
Microsoft Windows Platform SDK
```

- awk(awk, gawk)

Apache awk . exeawk(PerlWSH/VB)Brian
Kernighan <http://cm.bell-labs.com/cm/cs/who/bwk/Win32>
<http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe>awk . exe
awk95 . exe

```
Developer StudioTools - OptionsDirectories awk.exe(
Developer Studio 7.0 the Projects - VC++ Directories )
awk.exe PATH
```

```
Cygwin (http://www.cygwin.com/)awk gawk.exeawk.exe
gawk.exe Windowscygwin awk.exegawk.exe
awk.exe
```

- `[] OpenSSL(mod_sslab.exessl)`

OpenSSLOpenSSLApacheOpenSSL

```
mod\_ssllabs(ab.exeSSL)OpenSSL src\libopenssl
openSSL http://www.openssl.org/source/ releasedebug
0.9.7
```

```
perl Configure VC-WIN32
perl util\mkfiles.pl >MINFO
perl util\mk1mf.pl dll no-asm no-mdc2 no-rc5
no-idea VC-WIN32 >makefile
perl util\mk1mf.pl dll debug no-asm no-mdc2
no-rc5 no-idea VC-WIN32 >makefile.dbg
perl util\mkdef.pl 32 libeay no-asm no-mdc2
no-rc5 no-idea >ms\libeay32.def
perl util\mkdef.pl 32 ssleay no-asm no-mdc2
no-rc5 no-idea >ms\ssleay32.def
nmake
nmake -f makefile.dbg
```

- `[] zlib (mod_deflate)`

```
Zlibsrc\libzlib mod\_deflate Zlib
http://www.gzip.org/zlib/ -- mod\_deflate 1.1.4
```



Apache cd

Apache make Makefile.win Windows NT Apache release
debug

```
nmake /f Makefile.win _apacher
```

```
nmake /f Makefile.win _apached
```

Apache bugs



ApacheVC++Visual StudioVisual Studio Apache.dsw
Apache .dsp
Apache.dsw InstallBin (Release Debug)
InstallBin Makefile.win
GeneralBuild Command line INSTDIR /Ap
BuildBin

.dspVisual C++ 6.0Visual C++ 5.0 (97)Visual C++
Apache.dsw.dsp Apache.sln.msproj .dsp
VC++ 7.0 Apache.dsw

Visual C++ 7.0 (.net)Build Configuration Managerabs
mod_deflate DebugRelease srclibopensslzlib
nmakeBinBuild

.mak Visual C++ 5.0 mod_ssl abs(SSL
VC++ 7.0 (.net) nmake binenv VC++ 5.0
Project - Exportmake

```
perl srclib\apr\build\fixwin32mak.pl
```

httpd .mak .dep .dsp

Visual Studio 6.0 VC++



Apache.dswmakefile.win nmakeApache.dsp

1. srclib\apr\apr.dsp
2. srclib\apr\libapr.dsp
3. srclib\apr-util\uri\gen_uri_delims.dsp
4. srclib\apr-util\xml\expat\lib\xml.dsp
5. srclib\apr-util\aprutil.dsp
6. srclib\apr-util\libaprutil.dsp
7. srclib\pcre\dftables.dsp
8. srclib\pcre\pcre.dsp
9. srclib\pcre\pcreposix.dsp
10. server\gen_test_char.dsp
11. libhttpd.dsp
12. Apache.dsp

modules\

support\Apache

Apac

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

Apache \Apache2

dir nmake

```
nmake /f Makefile.win installr INSTDIR=dir
```

```
nmake /f Makefile.win installd INSTDIR=dir
```

INSTDIR*dir* \Apache2

- *dir*\bin\Apache.exe - Apache
- *dir*\bin\ApacheMonitor.exe -
- *dir*\bin\htdigest.exe - (Digest auth password datab
- *dir*\bin\htdbm.exe - SDBM(SDBM auth datab
- file utility)
- *dir*\bin\htpasswd.exe - (Basic auth password f
- *dir*\bin\logresolve.exe - dns
- *dir*\bin\rotatelogs.exe -
- *dir*\bin>wintty.exe -
- *dir*\bin\libapr.dll - Apache
- *dir*\bin\libaprutil.dll - Apache
- *dir*\bin\libhttpd.dll - Apache
- *dir*\modules\mod_*.so - Apache
- *dir*\conf -
- *dir*\logs -
- *dir*\include - C
- *dir*\lib -

Apache

.dsp

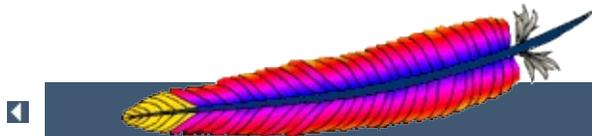
.mak

Developer Studio

```
makeBuildBin( _apacher _apachec
```

```
.mak .mak ( .dep)Platform SDK  
DevStudio\SharedIDE\bin\VC5)  
DevStudio\Common\MSDev98\bin\VC6) sysincl.dat  
VC++ (  
src\lib\apr\build\fixwin32mak.pl.mak
```

||||



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

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.



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 for 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 `ServerRootServerName` 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.CONVMAGIC` 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
```

```
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/magicconf/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 `NOVELLLIBC` 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\clidapsdk\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 source code directory.

```
Set AP_WORK=D:\httpd-2.0.x
```

- Set the environment variable APR_WORK to the full path of the apr source code directory. Typically \httpd\src\lib\apr but the APR project can be outside of the httpd directory structure.

```
Set APR_WORK=D:\apr-1.x.x
```

- Set the environment variable `APU_WORK` to the full path of the `apr-util` source code directory. Typically `\httpd\src\lib\apr-util` but the APR-UTIL project can be outside of the `httpd` directory structure.

```
Set APU_WORK=D:\apr-util-1.x.x
```

- 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` and build the prebuild utilities by running "`gmake -f nwgnumakefile prebuild`". This target will create the directory `\httpd-2.0\nwprebuild` and copy each of the utilities to this location that are necessary to complete the following build steps.
- Copy the files `\httpd-2.0\nwprebuild\GENCHARS.nlm` `\httpd-2.0\nwprebuild\DFTABLES.nlm` 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` `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 nwgnumakefile`
Builds release versions of all of the binaries and copies them to a `\release` destination directory.
- `gmake -f nwgnumakefile DEBUG=1`
Builds debug versions of all of the binaries and copies them to a `\debug` destination directory.
- `gmake -f nwgnumakefile install`
Creates a complete Apache distribution with binaries, docs and additional support files in a `\dist\Apache2` directory.
- `gmake -f nwgnumakefile prebuild`
Builds all of the prebuild utilities and copies them to the `\nwprebuild` directory.
- `gmake -f nwgnumakefile installdev`
Same as `install` but also creates a `\lib\include` directory in the destination directory and copies headers and import files.
- `gmake -f nwgnumakefile clean`
Cleans all object files and binaries from the `\release.o` `\debug.o` build areas depending on whether `DEBUG` has been defined.
- `gmake -f nwgnumakefile clobber_all`
Same as `clean` and also deletes the distribution directory if it exists.

Additional environment variable options

- To build all of the experimental modules, set the environment variable `EXPERIMENTAL`:

```
Set EXPERIMENTAL=1
```

- To build Apache using standard BSD style sockets rather than Winsock, set the environment variable USE_STDSOCKETS:

```
Set USE_STDSOCKETS=1
```

Building mod_ssl for the NetWare platform

By default Apache for NetWare uses the built-in module [mod_nw_ssl](#) to provide SSL services. This module simply enables the native SSL services implemented in NetWare OS to handle all encryption for a given port. Alternatively, mod_ssl can also be used in the same manner as on other platforms.

Before mod_ssl can be built for the NetWare platform, the OpenSSL libraries must be provided. This can be done through the following steps:

- Download the latest NetWare patch for OpenSSL from the [OpenSSL Contribution](#) page.
- Download the corresponding OpenSSL source code from the [OpenSSL Source](#) page.
- At the root of the OpenSSL source directory, apply the NetWare patch using the "patch" utility, for example:

```
patch -p 1 -i netwarepatch-0.9.7g.diff
```

- Edit the file NetWare/set_env.bat and modify any tools and utilities paths so that they correspond to your build environment.
- From the root of the OpenSSL source directory, run the following scripts:

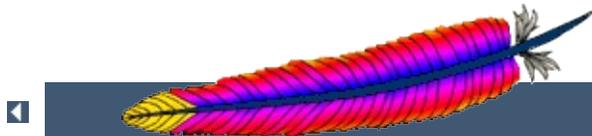
```
Netware/set_env netware-libc
```

```
Netware/build netware-libc
```

- Before building Apache, set the environment variable OSSLSDK to the full path to the root of the openssl source code directory.

```
Set OSSLSDK=d:\openssl-0.9.7x
```

|||



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Running a High-Performance Web Server on HPUX

Date: Wed, 05 Nov 1997 16:59:34 -0800
From: Rick Jones <raj@cup.hp.com>
Reply-To: raj@cup.hp.com
Organization: Network Performance
Subject: HP-UX tuning tips

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.

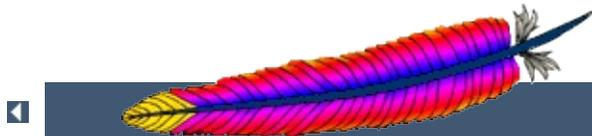
If folks are concerned about the number of FIN_WAIT_2 connections, they can use nettune to shrink the value of tcp_keepstart. However, they should be careful there - certainly do not make it less than oh two to four minutes. If tcp_hash_size has been set well, it is probably OK to let the FIN_WAIT_2's 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>



Apache HTTP Server 2.2

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

| | < > | ??? |

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 grepsed 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()` `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\015` which are ubiquitous in the server code are an exception: they are already the binary encoding of the ASCII `\n\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, .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.



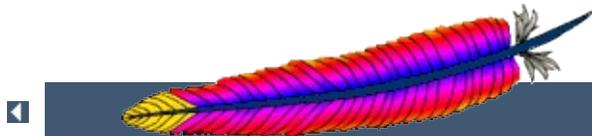
Module	Status	Notes
<u>core</u>	+	
<u>mod_authz_host</u>	+	
<u>mod_actions</u>	+	
<u>mod_alias</u>	+	
<u>mod_asis</u>	+	
<u>mod_auth_basic</u>	+	
<u>mod_authn_file</u>	+	
<u>mod_authn_anon</u>	+	
<u>mod_authn_dbm</u>	?	with own libdb.a
<u>mod_autoindex</u>	+	
<u>mod_cern_meta</u>	?	
<u>mod_cgi</u>	+	
mod_digest	+	
<u>mod_dir</u>	+	
<u>mod_so</u>	-	no shared libs
<u>mod_env</u>	+	
<u>mod_example</u>	-	(test bed only)
<u>mod_expires</u>	+	
<u>mod_headers</u>	+	
<u>mod_imagemap</u>	+	
<u>mod_include</u>	+	
<u>mod_info</u>	+	
mod_log_agent	+	
mod_log_config	+	
<u>mod_mime</u>	+	
<u>mod_mime_magic</u>	?	not ported yet
<u>mod_negotiation</u>	+	

<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
mod_jserv	-	JAVA still being ported.
mod_php3	+	mod_php3 runs fine, with LDAP and GD and FreeType libraries.
mod_put	?	untested
mod_session	-	untested



◀ **Apache HTTP Server 2.2**

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

| | | 2006114 |

httpd - Apache

httpdApache(HTTP)

httpdUnix [apachectl](#) [Windows NT/2000/XP/2003](#)
[Windows 95/98/ME](#).



```
httpd [ -d serverroot ] [ -f config ] [ -C
directive ] [ -c directive ] [ -D parameter ] [ -e
level ] [ -E file ] [ -k
start|restart|graceful|stop|graceful-stop ] [ -R
directory ] [ -h ] [ -l ] [ -L ] [ -S ] [ -t ] [ -v
] [ -V ] [ -X ] [ -M ]
```

[Windows](#)

```
httpd [ -k install|config|uninstall ] [ -n name ] [
-w ]
```



-d *serverroot*
ServerRootserverrootServerRoot /usr/local/apache2

-f *config*
config config"!" ServerRoot conf/httpd.conf

-k start|restart|graceful|stop|graceful-stop
httpd Apache

-C *directive*
directive

-c *directive*
directive

-D *parameter*
parameter <IfDefine>

-e *level*
LogLevellevel

-E *file*
file

-R *directory*
SHARED_CORE directory

-h

-l
LoadModule

-L

-M
DSO

-S
()

-t "0"(OK)0(Error)"-D *DUMP_VHOSTS*

-v httpd

-V httpd

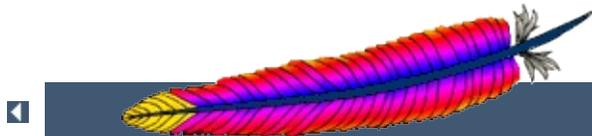
-X httpd

Windows

-k **install|config|uninstall**
ApacheWindows NTApacheApache

-n *name*
Apachename

-w



Apache HTTP Server 2.2

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

| | | 2006113 |

ab - Apache HTTP

abApache(HTTP)ApacheApache



```
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*
" : "base64(401)

-c *concurrency*

-C *cookie-name=value*
" Cookie:" *name=value*

-d
"percentage served within XX [ms] table")

-e *csv-file*
(CSV)(1%100%)()""gnuplot"

-g *gnuplot-file*
"gnuplot"TSV(Tab) Gnuplot, IDL, Mathematica, Excel

-h

-H *custom-header*
("Accept-Encoding: zip/zop;8bit")

-i
HEAD GET

-k
KeepAliveHTTPKeepAlive

-n *requests*

-p *POST-file*
POST

-P *proxy-auth-username:password*
" : "base64(407)

-q
150 ab10%100 stderr -q

-s
(ab -h)SSL https http

-S
12//()

-t *timelimit*
" -n 50000"

-T *content-type*
POST"Content-type"

-v *verbosity*
4 3(404200) 2

-V

-w
HTML

-x *<table>-attributes*
<table> <table >

-X *proxy[:port]*

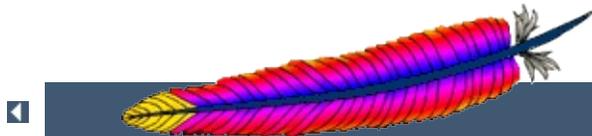
-y *<tr>-attributes*
<tr>

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



HTTP/1.x "" strstr() ab

||||



Apache HTTP Server 2.2

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

| | | 2006113 |

apachectl - Apache HTTP

apachectl Apache HTTP Apache

apachectl [httpd](#) [httpd](#) SysV
start, restart, stop [httpd](#)

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

apachectl >0



apachectl[httpd](#)

apachectl [*httpd-argument*]

SysV apachectl

apachectl *command*



SysV [httpd](#)

start

Apache [httpd](#) apachectl -k start

stop

Apache [httpd](#) apachectl -k stop

restart

Apache [httpd](#) configtestApache
apachectl -k restart

fullstatus

[mod_status](#) [mod_status](#) lynx STATUSURL
URL

status

fullstatus

graceful

Apache [httpd](#)
configtestApache apachectl -k graceful

graceful-stop

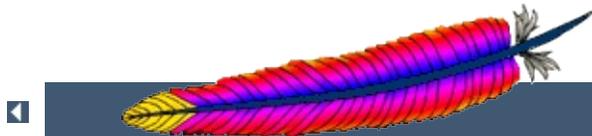
Apache [httpd](#)
stop

configtest

Syntax Ok apachectl -t

startssl

SSL[httpd](#) SSL apachectl start



Apache HTTP Server 2.2

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

| | | 2006113 |

apxs - Apache

apxs Apache HTTP [mod_so](#) [LoadModule](#) Apache

DSO Apache [httpd](#) [mod_so](#) apxs

```
$ httpd -l
```

[mod_so](#) apxs DSO Apache

```
$ apxs -i -a -c mod_foo.c
gcc -fpic -DSHARED_MODULE -
I/path/to/apache/include -c mod_foo.c
ld -Bshareable -o mod_foo.so mod_foo.o
cp mod_foo.so
/path/to/apache/modules/mod_foo.so
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module 'foo' in
/path/to/apache/etc/httpd.conf]
$ apachectl restart
/path/to/apache/sbin/apachectl restart: httpd
not running, trying to start
[Tue Mar 31 11:27:55 1998] [debug]
mod_so.c(303): loaded module foo_module
/path/to/apache/sbin/apachectl restart: httpd
started
$ _
```

*files*C(.c)(.o)(.a) apxsC(PIC)GCC -fpic

C apxs

Apache DSO [mod_so](#)
src/modules/standard/mod_so.c



apxs -g [**-S** *name=value*] **-n** *modname*

apxs -q [**-S** *name=value*] *query* ...

apxs -c [**-S** *name=value*] [**-o** *dsofile*] [**-I** *incdir*] [**-D** *name=value*] [**-L** *libdir*] [**-l** *libname*] [**-Wc,compiler-flags**] [**-Wl,linker-flags**] *files* ...

apxs -i [**-S** *name=value*] [**-n** *modname*] [**-a**] [**-A**] *dso-file* ...

apxs -e [**-S** *name=value*] [**-n** *modname*] [**-a**] [**-A**] *dso-file* ...



-n *modname*

-i() *-g()* *-g* *-i* *apxs()*

-q

apxs query CC, CFLAGS, CFLAGS_SHLIB, INCLUDEDIR, LD_SHLIB, LDFLAGS_SHLIB, LIBEXECDIR, LIBS_SHLIB, SBINDIR, SYSCONFDIR, TARGET ApacheCMakefile

```
INC=-I`apxs -q INCLUDEDIR`
```

-S *name=value*

apxs

-g

name(-n) mod_name.capxs Makefile

DSO

-c

C(.c) files(.o) files(.o.a) dsofile -o files mod_name.so

-o *dsofile*

files mod_unknown.so

-D *name=value*

-I *incdir*

-L *libdir*

-l *libname*

-Wc, compiler-flags

libtool --mode=compile *compiler-flags*

-Wl, linker-flags

libtool --mode=link *linker-flags*

DSO

-i

modules

-a

LoadModulehttpd.conf

-A

-a LoadModule(#)

-e

-a -A -i Apache httpd.conf



Apachemod_foo.c CApache

```
$ apxs -c mod_foo.c
/path/to/libtool --mode=compile gcc ... -c
mod_foo.c
/path/to/libtool --mode=link gcc ... -o mod_foo.la
mod_foo.slo
$ _
```

Apache [LoadModule](#) apxs"modules"
httpd.conf

```
$ apxs -i -a mod_foo.la
/path/to/instldso.sh mod_foo.la
/path/to/apache/modules
/path/to/libtool --mode=install cp mod_foo.la
/path/to/apache/modules ... chmod 755
/path/to/apache/modules/mod_foo.so
[activating module 'foo' in
/path/to/apache/conf/httpd.conf]
$ _
```

```
LoadModule foo_module modules/mod_foo.so
```

-A

```
$ apxs -i -A mod_foo.c
```

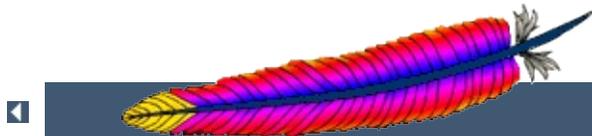
apxsApacheMakefile

```
$ apxs -g -n foo
```

```
Creating [DIR] foo
Creating [FILE] foo/Makefile
Creating [FILE] foo/modules.mk
Creating [FILE] foo/mod_foo.c
Creating [FILE] foo/.deps
$ _
```

Apache

```
$ cd foo
$ make all reload
apxs -c mod_foo.c
/path/to/libtool --mode=compile gcc ... -c
mod_foo.c
/path/to/libtool --mode=link gcc ... -o mod_foo.la
mod_foo.slo
apxs -i -a -n "foo" mod_foo.la
/path/to/inststdso.sh mod_foo.la
/path/to/apache/modules
/path/to/libtool --mode=install cp mod_foo.la
/path/to/apache/modules ... chmod 755
/path/to/apache/modules/mod_foo.so
[activating module 'foo' in
/path/to/apache/conf/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
$ _
```



◀ **Apache HTTP Server 2.2**

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

| | | 2006115 |

configure -

configureApacheApache

Unix



configure

./configure [*OPTION*]... [*VAR=VALUE*]...

(CC, CFLAGS ...) VAR=VALUE



-
- -
 -
 -
 -
 - [apr-config](#)
 -
 -

configure"[]"

-C

--config-cache

 --cache-file=config.cache

--cache-file=FILE

 FILE()

-h

--help [short|recursive]

 shortApache recursive

-n

--no-create

 configure

-q

--quiet

 " checking ..."

--srcdir=DIR

 DIR [configure]

--silent

`--quiet`

`-V`

`--version`

`"[]"`

`--prefix=PREFIX`

`PREFIX Apache [/usr/local/apache2]`

`--exec-prefix=EPREFIX`

`EPREFIX [PREFIX]`

`make install /usr/local/apache2/bin ,
/usr/local/apache2/lib --prefix
/usr/local/apache2 --prefix=$HOME`

`--enable-layout=LAYOUT`

`LAYOUT Apache config.layout <Layout
FOO>...</Layout> FOO Apache`

`autoconf"[]"`

`--bindir=DIR`

`DIR htpasswd, dbmmanage
[EPREFIX/bin]`

`--datadir=DIR`

`WebDIR autoconfApache
[PREFIX/share]`

`--includedir=DIR`

ApacheCDIR
[EPREFIX/include]

--infodir=DIR
DIR autoconfApache
[PREFIX/info]

--libdir=DIR
DIR
[EPREFIX/lib]

--libexecdir=DIR
DIR
[EPREFIX/libexec]

--localstatedir=DIR
DIR autoconfApache
[PREFIX/var]

--mandir=DIR
DIR
[EPREFIX/man]

--oldincludedir=DIR
gccCDIR autoconfApache
[/usr/include]

--sbindir=DIR
DIR HTTP[httpd](#), [apachectl](#), [suexec](#)
[EPREFIX/sbin]

--sharedstatedir=DIR
DIR autoconfApache
[PREFIX/com]

--sysconfdir=DIR
DIR httpd.confmime.types
[PREFIX/etc]

Apache HTTPApache HTTP"[]"

--build=*BUILD*

BUILD

[config.guess]

--host=*HOST*

Apache HTTP*HOST*

[*BUILD*]

--target=*TARGET*

configure for building compilers for *TARGET* autoconf

Apache

[*HOST*]

DSODSOmod_soDSODSO"--enable-
so=static"

--disable-*MODULE*

MODULE()

--enable-*MODULE*=shared

*MODULE*DSO()

--enable-*MODULE*=static

MODULE()

--enable-mods-shared=*MODULE-LIST*

*MODULE-LIST*DSO()

--enable-modules=*MODULE-LIST*

MODULE-LIST()

MODULE-LIST

(1)

```
--enable-mods-shared='headers rewrite dav'
```

(2)"most"() (3)" all"()

```
--enable-mods-shared=most
```

```
configureMODULEMODULE-LIST      MODULEMODULE-  
LIST' mod_NAME"" mod_"" _"" -"  
" mod_log_config"" log-config"
```

(B)(E)/(X)

<u>mod actions</u>	(B)	CGI
<u>mod alias</u>	(B)	URL
<u>mod asis</u>	(B)	HTTP
<u>mod auth basic</u>	(B)	
<u>mod authn default</u>	(B)	
<u>mod authn file</u>	(B)	
<u>mod authz default</u>	(B)	
<u>mod authz groupfile</u>	(B)	
<u>mod authz host</u>	(B)	IP
<u>mod authz user</u>	(B)	
<u>mod autoindex</u>	(B)	"ls""dir"
<u>mod cgi</u>	(B)	MPM(<u>prefork</u>)CGI
<u>mod cgid</u>	(B)	MPM(<u>worker</u>)CGICGI

<u>mod_dir</u>	(B)	""
<u>mod_env</u>	(B)	ApacheCGISSI
<u>mod_filter</u>	(B)	
<u>mod_imagemap</u>	(B)	
<u>mod_include</u>	(B)	(SSI)
<u>mod_isapi</u>	(B)	WindowsISAPI
<u>mod_log_config</u>	(B)	
<u>mod_mime</u>	(B)	(/)(MIME//)
<u>mod_negotiation</u>	(B)	
<u>mod_nw_ssl</u>	(B)	NetWareSSL
<u>mod_setenvif</u>	(B)	
<u>mod_status</u>	(B)	Web
<u>mod_userdir</u>	(B)	("/~username")
<u>mod_auth_digest</u>	(X)	MD5()
<u>mod_authn_alias</u>	(E)	
<u>mod_authn_anon</u>	(E)	
<u>mod_authn_dbd</u>	(E)	SQL
<u>mod_authn_dbm</u>	(E)	DBM
<u>mod_authnz_ldap</u>	(E)	LDAP
<u>mod_authz_dbm</u>	(E)	DBM
<u>mod_authz_owner</u>	(E)	
<u>mod_cache</u>	(E)	URI()
<u>mod_cern_meta</u>	(E)	ApacheCERN httpd
<u>mod_charset_lite</u>	(X)	
<u>mod_dav</u>	(E)	ApacheDAV
<u>mod_dav_fs</u>	(E)	<u>mod_dav</u>
<u>mod_dav_lock</u>	(E)	<u>mod_dav</u>
<u>mod_dbd</u>	(E)	SQL
<u>mod_deflate</u>	(E)	

<u>mod_disk_cache</u>	(E)	
<u>mod_dumpio</u>	(E)	I/O
<u>mod_echo</u>	(X)	
<u>mod_example</u>	(X)	ApacheAPI
<u>mod_expires</u>	(E)	HTTP" Expires:"" Cache-Control:"
<u>mod_ext_filter</u>	(E)	
<u>mod_file_cache</u>	(X)	Apache
<u>mod_headers</u>	(E)	HTTP
<u>mod_ident</u>	(E)	RFC1413ident
<u>mod_info</u>	(E)	ApacheWeb
<u>mod_ldap</u>	(E)	LDAPLDAP
<u>mod_log_forensic</u>	(E)	""
<u>mod_logio</u>	(E)	/HTTP
<u>mod_mem_cache</u>	(E)	
<u>mod_mime_magic</u>	(E)	MIME
<u>mod_proxy</u>	(E)	HTTP/1.1/
<u>mod_proxy_ajp</u>	(E)	<u>mod_proxy</u> Apache JServ Protocol
<u>mod_proxy_balancer</u>	(E)	<u>mod_proxy</u>
<u>mod_proxy_connect</u>	(E)	<u>mod_proxy</u> HTTP CONNECT
<u>mod_proxy_ftp</u>	(E)	<u>mod_proxy</u> FTP
<u>mod_proxy_http</u>	(E)	<u>mod_proxy</u> HTTP
<u>mod_rewrite</u>	(E)	URL
<u>mod_so</u>	(E)	DSO
<u>mod_speling</u>	(E)	URL
<u>mod_ssl</u>	(E)	(SSL)(TLS)
<u>mod_suexec</u>	(E)	<u>webCGISSI</u>
<u>mod_unique_id</u>	(E)	

mod_usertrack	(E)	Session(Cookie)
mod_version	(E)	
mod_vhost_alias	(E)	

(MPM)

[MPM](#)

--with-mpm=MPM

MPM [MPM](#)MPM [beos](#), [mpmt_os2](#), [prefork](#),
[worker](#)

--with-module=*module-type:module-file*[, *module-type:module-file*]

```
module-fileApache"      modules/module-type"
configuremodule-file " modules/module-type"
" modules/module-type"      configure
" modules/module-type" Makefile.in
```

- 1.
2. DSO

[apxs](#)(Apache)

--enable-nonportable-atomics

486CPUApache

--enable-v4-mapped

IPv4IPv6FreeBSDNetBSDOpenBSD

--disable-v4-mapped

IPv4IPv6FreeBSDNetBSDOpenBSD

--enable-maintainer-mode

--enable-exception-hook

EnableExceptionHandler

--with-port=PORT

httpd[80] httpd.conf

--with-program-name=NAME

[httpd]" NAME.conf"

apr-config

--disable-threads

MPM

--disable-ipv6

IPv6

--disable-dso

DSO

--with-apr=DIR|FILE

Apache(APR)httpdhttpdAPR apr-configAPR(apr-
config" bin")

--with-apr-util=DIR|FILE

Apache(APU)httpdhttpdAPU apu-configAPU(apu-
config" bin")

--with-ssl=DIR

mod_ssl configureOpenSSL SSL/TLS

--with-z=DIR

([mod deflate](#)) configurezlib

--with-perl=DIR

Perl [apxsdbmmanage](#) Perl5(5.003)PerlPerl 4 Perl 5
Perl 5Apache httpd

--with-pcre=DIR

5.0Perl(PCRE)PCRE

--with-ldap=DIR

Apache [mod ldapmod authnz ldap](#)APULDAP()
LDAP

Apache [mod authn_dbmmod rewrite](#)DBMAPUSDBM

--with-gdbm[=path]

GNU DBMSDBM *path* configureGNU DBM *path*
configure*path/libpath/include*GNU DBM" *inc-*
path:lib-path"GNU DBM

--with-ndbm[=path]

New DBMSDBM *path* configureNew DBM *path*
configure*path/libpath/include*New DBM" *inc-*
path:lib-path"New DBM

--with-berkeley-db[=path]

Berkeley DBSDBM *path* configureBerkeley DB *path*
configure*path/libpath/include*Berkeley DB" *inc-*
path:lib-path"Berkeley DB

DBMAPUAPU

--with-apr-util APUDBM

- enable-static-support**
()
- enable-static-ab**
[ab](#)
- enable-static-checkgid**
checkgid
- enable-static-htdbm**
[htdbm](#)
- enable-static-htdigest**
[htdigest](#)
- enable-static-htpasswd**
[htpasswd](#)
- enable-static-logresolve**
[logresolve](#)
- enable-static-rotatelog**
[rotatelog](#)

suexec

- enable-suexec**
[suexec](#) CGLuidgid **suexec**

[suexec](#)"[]" [suEXEC](#)
- with-suexec-bin**
[suexec](#)[- -sbindir]
- with-suexec-caller**
[suexec](#) [httpd](#)
- with-suexec-docroot**
[suexec](#)[- -datadir/htdocs]
- with-suexec-gidmin**
[suexec](#)GID[100]

--with-suexec-logfile

`suexec[suexec_log --logfiledir]`

--with-suexec-safepath

`suexec"" PATH[/usr/local/bin:/usr/bin:/bin]`

--with-suexec-userdir

`suexec suexec(mod_userdir)[public_html]`

--with-suexec-uidmin

`suexecUID[100]`

--with-suexec-umask

`suexecumask[]`



configure configure/

CC

C

CFLAGS

Cflags

CPP

C

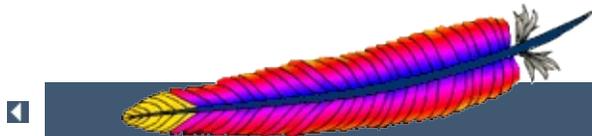
CPPFLAGS

C/C++flags" -I*includedir* *includedir*

LDFLAGS

flags"-L -L*libdir* *libdir*

||||



Apache HTTP Server 2.2

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

| | | 2006113 |

dbmmanage - DBM

dbmmanageDBM [mod_authn_dbm](#)HTTPApache HTTP
dbmmanageDBM [htpasswd](#)



```
dbmmanage [ encoding ] filename  
add|adduser|check|delete|update username [  
encpasswd [ group[,group...] [ comment ] ] ]
```

```
dbmmanage filename view [ username ]
```

```
dbmmanage filename import
```



filename

DBM .db, .pag, .dir

username

username(:)

encpasswd

updateadd(-) update(.)

group

(:)(-) *comment* update(.)

comment**-d**

crypt (WindowsNetware)

-m

MD5 (WindowsNetware)

-s

SHA1

-p

()

add

filenameusernameencpasswd

```
dbmmanage passwords.dat add rbowen  
foKntnEF3KSXA
```

adduser

filenameusername

```
dbmmanage passwords.dat adduser krietz
```

check

filenameusername

```
dbmmanage passwords.dat check rbowen
```

delete

filenameusername

```
dbmmanage passwords.dat delete rbowen
```

import

STDIN *username:password () filename*

update

adduser *usernamefilename*

```
dbmmanage passwords.dat update rbowen
```

view

DBM *username*

```
dbmmanage passwords.dat view
```



DBMSDBM, NDBM, GDBM, Berkeley DB 2

filenamedbmmanage dbmmanageDBMnothing DBM
DBM

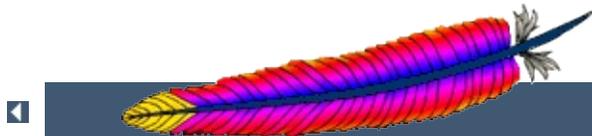
dbmmanageDBM @AnyDBM::ISA Berkeley DB 2

dbmmanageBerkeley DB 2, NDBM, GDBM, SDBM

dbmmanageDBMperl @AnyDBM::ISA DBMC

Unix fileDBM

||||



Apache HTTP Server 2.2

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

| | | 2006113 |

htcacheclean -

htcacheclean mod disk cache TERM INT



```
htcacheclean [ -D ] [ -v ] [ -t ] [ -r ] [ -n ] -  
ppath -llimit
```

```
htcacheclean -b [ -n ] [ -t ] [ -i ] -dinterval -  
ppath -llimit
```



-dinterval

interval -D, -v, -r SIGTERMSIGINT

-D

"" -d

-v

-d

-r

Apache web() -d -t

-n

htcacheclean(a)IO (b)

-t

inode

-ppath

path CacheRoot

-llimit

limit xxBxx xxKxx xxMxx

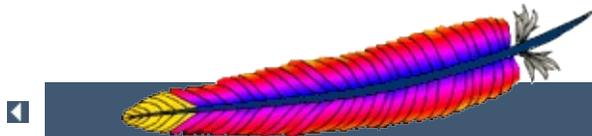
-i

-d



htcacheclean" 0"" 1"

||||



Apache HTTP Server 2.2

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

| | | 2006321 |

htdbm - DBM

htdbm mod_authn_dbm HTTPDBM dbmmanage 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

-c

passwdfile passwdfile -n

-n

passwdfile() -c

-m

MD5Windows, Netware, TPF

-d

crypt ()Windows, Netware, TPF ht dbmWindows,
Netware, TPF [httpd](#)

-s

SHA LDAPNetscape server

-p

() ht dbm [httpd](#)Windows, Netware, TPF

-l

-t

"Comment"

-v

"3"

-x

filename

DBM .db, .pag, .dir -c DBM

username

passwdfile username

password

-b

-TDBTYPE

DBM(SDBM, GDBM, DB, "default")



DBMSDBM, NDBM, GNU GDBM, Berkeley/Sleepycat DB 2/3/4

*filename*tdbm

|

DBM

Unix fileDBM



```
htdbm" 0"" 1"" 2"" 3"" 4"(username, filename,  
password,)" 5"( )" 6"" 7"
```



```
htdbm /usr/local/etc/apache/.htdbm-users jsmith
```

```
jsmithWindowsApacheMD5          crypt()      htdbm
```

```
htdbm -c /home/does/public_html/.htdbm jane
```

```
jane          htdbm
```

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

```
(Pwd4Steve)MD5
```



Web(htdbm)

-b

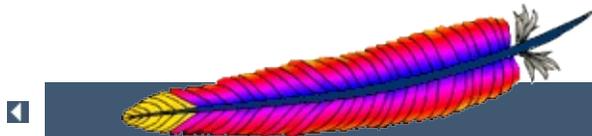


WindowsMPE htdbm255

htdbmMD5ApacheApacheWeb

255(:)

||||



Apache HTTP Server 2.2

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

| | | 2006114 |

htdigest -

htdigest// htdigest

mod_auth_digest



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



-c

passwdfile passwdfile

passwdfile

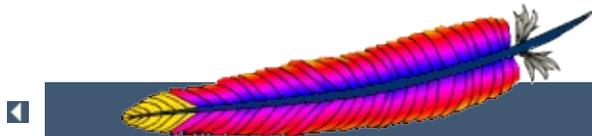
// -c

realm

username

passwdfile username

| | | |



Apache HTTP Server 2.2

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

| | | 2006114 |

htpasswd -

htpasswd/ htpasswd

htpasswdDBM [dbmmanage](#)

htpasswdApacheMD5crypt() htpasswdMD5
crypt()

[mod_auth_basic](#)



```
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

-c

passwdfile passwdfile -n

-n

Apache *passwdfile()* -c

-m

MD5Windows, Netware, TPF

-d

crypt ()Windows, Netware, TPF htpasswdWindows,
Netware, TPF [httpd](#)

-s

SHA LDAPNetscape server

-p

() htpasswd [httpd](#)Windows, Netware, TPF

-D

usernamepasswdfile

passwdfile

-C

username

passwdfile username

password

-b



htpasswdpasswordfile" 0"" 1"" 2"" 3"" 4"(username,
filename, password,)" 5"()" 6"" 7"



```
htpasswd /usr/local/etc/apache/.htpasswd-users  
jsmith
```

```
jsmithWindowsApacheMD5      crypt()  
    htpasswd
```

```
htpasswd -c /home/does/public_html/.htpasswd jane
```

```
jane      htpasswd
```

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

```
(Pwd4Steve)MD5
```



Web(htpasswd)

-b

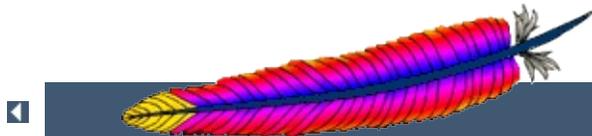


WindowsMPE htdbm255

htdbmMD5ApacheApacheWeb

255(:)

||||



Apache HTTP Server 2.2

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

| | | 2006114 |

logresolve - ApacheIP

logresolveApachePIP

ApacheIP



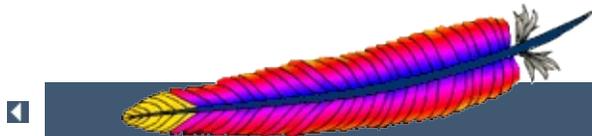
```
logresolve [ -s filename ] [ -c ] < access_log >  
access_log.new
```



-s *filename*

-c
logresolveDNSIP

||||



Apache HTTP Server 2.2

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

| | | 2006114 |

rotatelogs - Apache

rotatelogsApache

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

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

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

5

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

5 errorlog.YYYY-mm-dd-HH_MM_SS



```
rotatelogs [ -1 ] logfile [ rotationtime [ offset  
] ] | [ filesizeM ]
```



-1

GMTGMT() -1

logfile

logfile"% strptime()" .nnnnnnnnnn"

rotationtime

offset

UTC"0"UTCUTC"-5"" -300"

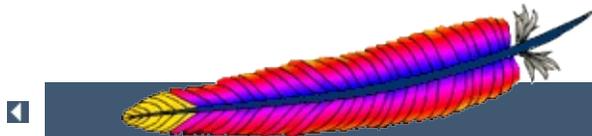
filesizeM

filesizeM



strftime() strftime()

%A	()
%a	3()
%B	()
%b	3()
%c	()
%d	2
%H	2(24)
%I	2(12)
%j	3
%M	2
%m	2
%p	am/pm12()
%S	2
%U	2()
%W	2()
%w	1()
%X	()
%x	()
%Y	4
%y	2
%Z	
%%	"%"



◀ **Apache HTTP Server 2.2**

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

| | | 2006114 |



Apache"

support"

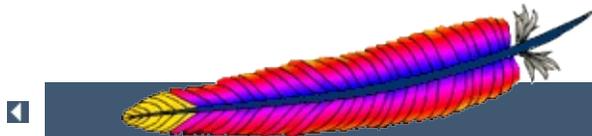


perlcron



perlweb(" %v")+ ".log"

webstdin



Apache HTTP Server 2.2

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

| | | 200619 |

Apache

Apache

- <http://purl.org/NET/http-errata> - HTTP/1.1
- <http://www.rfc-editor.org/errata.html> - RFC
- <http://ftp.ics.uci.edu/pub/ietf/http/#RFC> - HTTPRFC



ApachewebIETF

[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.



(HTML)ApacheIETFW3C

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.



ApacheIETF

RFC 2617 (Draft standard)

"HTTP/1.0", includes the specification for a Basic Access Authentication scheme.



ISO/

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

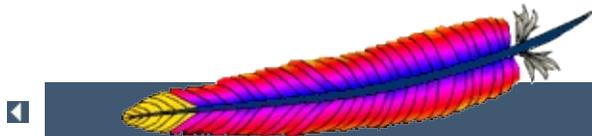
These pages document the country names (official short names in English) in alphabetical order as given in ISO 3166-1 and the corresponding ISO 3166-1-alpha-2 code elements.

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.

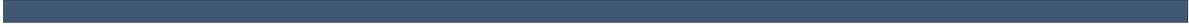


Apache HTTP Server 2.2

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

| | | 2006121 |





(Status)Apache

MPM

MPM

Base

Extension

Apache

Experimental

Apache

External

Apache(“")

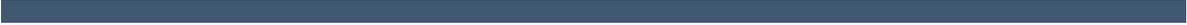


<IfModule>



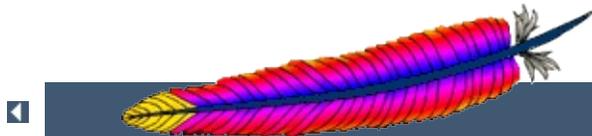
LoadModule





Apache2.0





Apache HTTP Server 2.2

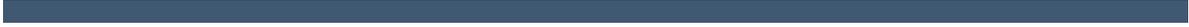
[Apache](#) > [HTTP Server](#) > > [2.2](#) >

| | | 2006121 |



Apache





0"l"

"..."

URL

http://www.example.com/path/to/file.html

URL-path

URL" /path/to/file.html")

file-path

" /usr/local/apache/htdocs/path/to/file.html"
(/) [ServerRoot](#)

directory-path

/usr/local/apache/htdocs/path/to/

filename

file.html

regex

Perl regex

extension

filename".Apache extension filename"."."
"." extension " file.html.en" extension .h
Apache extension"."

MIME-type

text/html

env-variable

Apache



(Apache)"

None"httpd.conf



server config

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

virtual host

<VirtualHost>

directory

<Directory>, <Location>, <Files>, <Proxy>

.htaccess

.htaccess overrides

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



.htaccess .htaccess

AllowOverride()

AllowOverride



Apache

Core

Apache

MPM

MPM

Base

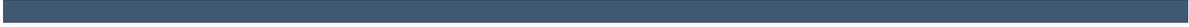
Apache

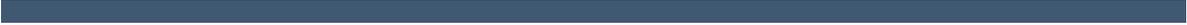
Extension

Apache

Experimental

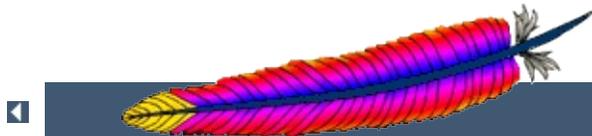






Apache2





◀ Apache HTTP Server 2.2

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

| | | 2006120 |

Apache(Core)

Apache HTTP

(C)



Accept Filter

Socket
AcceptFilter <i>protocol accept_filter</i>
server config
(C)
core
Apache 2.1.5

socketHTTPsocket [FreeBSD\(Accept Filter\)](#)Linux(more primitive)TCP_DEFER_ACCEPT

FreeBSD

```
AcceptFilter http httpready  
AcceptFilter https dataready
```

httpready(Accept Filter)HTTP [accf_http\(9\)](#)HTTPS
[accf_data\(9\)](#)

Linux

```
AcceptFilter http data  
AcceptFilter https data
```

LinuxTCP_DEFER_ACCEPThttp noneTCP_DEFER_ACCEPT [tcp\(7\)](#)

none(accept filter) nntp

```
AcceptFilter nntp none
```



AcceptPathInfo

AcceptPathInfo On Off Default
AcceptPathInfo Default
server config, virtual host, directory, .htaccess
FileInfo
(C)
core
Apache 2.0.30

() PATH_INFO

```
/test/ here.html /test/here.html/more  
/test/nothere.html/morePATH_INFO" /more"
```

AcceptPathInfo

off

```
/test/here.html/more"404 NOT FOUND"
```

On

```
/test/here.html /test/here.html/more
```

Default

```
PATH_INFO cgi-scriptisapi-isaPATH_INFO
```

```
AcceptPathInfoPATH_INFO
```

```
INCLUDESPATH_INFO
```

```
<Files "mypaths.shtml">  
  Options +Includes  
  SetOutputFilter INCLUDES  
  AcceptPathInfo On  
</Files>
```



AccessFileName <i>filename</i>
AccessFileName .htaccess
server config, virtual host
(C)
core

```
AccessFileName .acl
```

```
/usr/local/web/index.html    /.acl /usr/.acl  
/usr/local/.acl /usr/local/web/.acl
```

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

- [AllowOverride](#)
-
- [.htaccess](#)



ADDDEFAULTCHARSET

text/plain text/html HTTP
AddDefaultCharset On Off <i>charset</i>
AddDefaultCharset Off
server config, virtual host, directory, .htaccess
FileInfo
(C)
core

```
text/plain|text/html|HTTP      <meta>
  AddDefaultCharset Off      AddDefaultCharset On
Apache          iso-8859-1 IANAcharset
```

```
AddDefaultCharset utf-8
```

AddDefaultCharset(CGI)

- [AddCharset](#)



AddOutputFilterByType

MIME
AddOutputFilterByType <i>filter</i> [<i>;filter...</i>] <i>MIME-type</i> [<i>MIME-type</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(C)
core
Apache 2.0.33 Apache 2.1

[MIME](#) [mod filter](#)

[mod deflate](#) DEFLATE text/html text/plain()

```
AddOutputFilterByType DEFLATE text/html text/plain
```

```
(;)      AddOutputFilterByType
```

```
text/html INCLUDES DEFLATE
```

```
<Location /cgi-bin/>  
  Options Includes  
  AddOutputFilterByType INCLUDES;DEFLATE  
  text/html  
</Location>
```

```
AddOutputFilterByType      MIME      DefaultType  
                                 DefaultType  
AddTypeForceType(non-nph)CGI
```

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



URL
AllowEncodedSlashes On Off
AllowEncodedSlashes Off
server config, virtual host
(C)
core
Apache 2.0.46

AllowEncodedSlashesURL("%2F" → "/" %5C" → "\\")URL
"404")

AllowEncodedSlashes On PATH_INFO

() %2F%5C()URL

- [AcceptPathInfo](#)



.htaccess
AllowOverride All None <i>directive-type</i> [<i>directive-type</i>] ...
AllowOverride All
directory
(C)
core

.htaccess([AccessFileName](#))

<Directory>

[AllowOverride<Directory>](#) [<Location>](#),
[<DirectoryMatch>](#), [<Files>](#)

None [.htaccess](#) .htaccess

All".htaccess" .htaccess

directive-type

AuthConfig

([AuthDBMGroupFile](#), [AuthDBMUserFile](#), [AuthGroupFile](#),
[AuthName](#), [AuthType](#), [AuthUserFile](#), [Require](#),)

FileInfo

([DefaultType](#), [ErrorDocument](#), [ForceType](#),
[LanguagePriority](#), [SetHandler](#), [SetInputFilter](#),
[SetOutputFilter](#), [mod_mime](#) Add* Remove*)
([Header](#), [RequestHeader](#), [SetEnvIf](#), [SetEnvIfNoCase](#),
[BrowserMatch](#), [CookieExpires](#), [CookieDomain](#),
[CookieStyle](#), [CookieTracking](#),
[CookieName](#)) [mod_rewrite](#)([RewriteEngine](#),
[RewriteOptions](#), [RewriteBase](#), [RewriteCond](#),

RewriteRule)mod_actionsAction

Indexes

(AddDescription, AddIcon, AddIconByEncoding,
AddIconByType, DefaultIcon, DirectoryIndex,
FancyIndexing, HeaderName, IndexIgnore,
IndexOptions, ReadmeName,)

Limit

(Allow, Deny, Order)

Options[=Option,...]

(OptionsXBitHack()) Options Options

.htaccessAuthConfigIndexes

AllowOverride AuthConfig Indexes

- AccessFileName
-
- .htaccess



AuthName

HTTP
AuthName <i>auth-domain</i>
directory, .htaccess
AuthConfig
(C)
core

AuthName AuthTypeRequireAuthUserFile
AuthGroupFile

AuthName "Top Secret"

AuthName

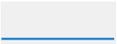
-
- ▣

AuthType

AuthType Basic Digest
directory, .htaccess
AuthConfig
(C)
core

Basic(mod_auth_basic)Digest(mod_auth_digest)

AuthNameRequire(mod_authn_file)(
mod_authz_user)

- 



CGIMapExtension

CGI
CGIMapExtension <i>cgi-path .extension</i>
directory, .htaccess
FileInfo
(C)
core
NetWare only

Apache CGI" CGIMapExtension sys:\foo.nlm .foo"
.fooCGIFOO



Content-Digest

Content-MD5
ContentDigest On Off
ContentDigest Off
server config, virtual host, directory, .htaccess
Options
(C)
core

RFC1854 RFC2068 Content-MD5

MD5""("")

Content-MD5

Content-MD5: AuLb7Dp1rqtRtxz2m9kRpA==

()

Content-MD5 [Apache](#)SSI CGI



DefaultType

MIME
DefaultType <i>MIME-type</i>
DefaultType text/plain
server config, virtual host, directory, .htaccess
FileInfo
(C)
core

MIME

DefaultType

DefaultType image/gif

gif.gif

ForceTypemimemime



Directory

<code><Directory <i>directory-path</i>> ... </Directory></code>
server config, virtual host
(C)
core

```
<Directory></Directory>"directory"           Directory-path
Unix shell"                                   ?""    *""
/*/public_html>/home/user/public_html <Directory
/home/*/public_html>
```

```
<Directory /usr/local/httpd/htdocs>
  Options Indexes FollowSymLinks
</Directory>
```

```
directory-pathApache    <Directory>
```

" ~"

```
<Directory ~ "^/www/(.+)*[0-9]{3}">
```

/www/3

```
() <Directory>()    .htaccess
```

```
<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/dir/.htaccessFileInfo

```
<Directory ~ abc$>  
  # .....  
</Directory>
```

`<Directory>.htaccess` /home/abc/public_html/abc

Apache `<Directory />" Allow from All"ApacheURL`

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

`<Directory>httpd.conf` `<Directory>` `<Limit>`
`<LimitExcept>`

- `<Directory><Location><Files>`



DirectoryMatch

<code><DirectoryMatch regex> ... </DirectoryMatch></code>
server config, virtual host
(C)
core

`<DirectoryMatch></DirectoryMatch>` [<Directory>](#)

```
<DirectoryMatch "^.*/www/(.+)/*[0-9]{3}">
```

`/www/3`

- [<Directory><Directory>](#)
- [<Directory><Location><Files>](#)



DocumentRoot <i>directory-path</i>
DocumentRoot /usr/local/apache2/htdocs
server config, virtual host
(C)
core

[httpd](#) [AliasURL](#) DocumentRoot

DocumentRoot /usr/web

http://www.my.host.com/index.html
 /usr/web/index.html *directory-path* [ServerRoot](#)

[DocumentRoot"/](#)

- [URL](#)



(memory-mapping)
EnableMMAP On Off
EnableMMAP On
server config, virtual host, directory, .htaccess
FileInfo
(C)
core

[httpd](#) [mod_include](#)Apache

- [httpd](#)
- NFS[DocumentRoot](#) [httpd](#)

EnableMMAP Off

NFS

```
<Directory "/path-to-nfs-files">
  EnableMMAP Off
</Directory>
```



sendfile
EnableSendfile On Off
EnableSendfile On
server config, virtual host, directory, .htaccess
FileInfo
(C)
core
Apache 2.0.44

[httpd](#)sendfile()Apachesendfile

sendfile

- sendfilesendfile
- LinuxIPv6sendfileTCPbug
- LinuxItaniumsendfile2GB
- NFS[DocumentRoot](#) (NFSSMB)

sendfile

```
EnableSendfile Off
```

NFSSMB

```
<Directory "/path-to-nfs-files">
  EnableSendfile Off
</Directory>
```



<code>ErrorDocument <i>error-code document</i></code>
server config, virtual host, directory, .htaccess
FileInfo
(C)
core
Apache2.0

Apache

- 1.
- 2.
3. *URL-path()*
4. *URL()*

12-4 `ErrorDocument` HTTP URL Apache/

`URL(/)URL(DocumentRoot)URL`

```
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"          default"Apache
ErrorDocument
```

```
ErrorDocument 404 /cgi-bin/bad_urls.pl
<Directory /web/docs>
```

```
ErrorDocument 404 default
</Directory>
```

```
    ErrorDocumentURL("    http")Apache
URLweb"
" ErrorDocument 401"
```

Microsoft Internet Explorer (MSIE)""""512 byte
MSIE



```
    ErrorDocument""
```

2.0



ErrorLog

ErrorLog <i>file-path</i> syslog[: <i>facility</i>]
ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows OS/2)
server config, virtual host
(C)
core

ErrorLog *file-path()* ServerRoot

```
ErrorLog /var/log/httpd/error_log
```

file-path()

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

```
"    syslog"syslogd(8)    local7 " syslog:facility"  
    facilitysyslog(1)
```

```
ErrorLog syslog:user
```

```
Unix(/)\
```

- [LogLevel](#)
- [Apache](#)



FileETag

ETag
FileETag <i>component</i> ...
FileETag INode MTime Size
server config, virtual host, directory, .htaccess
FileInfo
(C)
core

```
FileETag ETag() ( ETag) Apache1.3.22 ETag inode()
FileETag()
```

INode

(inode)

MTime

Size

All

```
FileETag INode MTime Size
```

None

ETag

```
INode, MTime, Size" +" -"
```

```
" FileETag INode MTime Size" FileETag -INode")
```

```
" FileETag MTime Size"
```



<code><Files filename> ... </Files></code>
server config, virtual host, directory, .htaccess
All
(C)
core

```
<Files> <Directory><Location> </Files>()
  <Files> <Directory>.htaccess <Location>
  <Files><Directory>
```

filename " ?"" *"" ~"

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

Apache1.3 <FilesMatch>

```
<Directory><Location> <Files>.htaccess
```

- [<Directory><Location><Files>](#)



FilesMatch

<code><FilesMatch regex> ... </FilesMatch></code>
server config, virtual host, directory, .htaccess
All
(C)
core

`<FilesMatch>`[<Files>](#)

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

internet

- [<Directory><Location><Files>](#)



ForceType

MIME
ForceType <i>MIME-type</i> None
directory, .htaccess
FileInfo
(C)
core
Apache 2.0

`.htaccess`
`<Directory>``<Location>``<Files>` *MIME-type*
Content-TypeGIF"

```
ForceType image/gif
```

DefaultTypemime

" None" ForceType

```
# image/gif:  
<Location /images>  
    ForceType image/gif  
</Location>  
  
# mime:  
<Location /images/mixed>  
    ForceType None  
</Location>
```



Hostname Lookups

IPDNS
HostnameLookups On Off Double
HostnameLookups Off
server config, virtual host, directory
(C)
core

DNS(REMOTE_HOSTCGI/SSI) DoubleDNSip
("tcpwrappers" PARANOID)

mod_authz_host HostnameLookups Double"
" HostnameLookups On"CGI REMOTE_HOS

Off off DNS binlogresolveIP



<code><IfDefine [!]parameter-name> ... </IfDefine></code>
server config, virtual host, directory, .htaccess
All
(C)
core

`<IfDefine test>...</IfDefine>` `<IfDefine>test` `test`

`<IfDefine>test`

- `parameter-name`
- `!parameter-name`

`parameter-name` `parameter-name`

`parameter-name` `httpd` `-Dparameter`

`<IfDefine>`

```
httpd -DReverseProxy ...
```

```
# httpd.conf
```

```
<IfDefine ReverseProxy>
```

```
  LoadModule rewrite_module
```

```
  modules/mod_rewrite.so
```

```
  LoadModule proxy_module modules/libproxy.so
```

```
</IfDefine>
```



MODULE

<code><IfModule [!]module-file module-identifier> ... </IfModule></code>
server config, virtual host, directory, .htaccess
All
(C)
core
<i>module-identifier</i> Apache 2.1

`<IfModule test>...</IfModule>` `<IfModule>test` `test`

`<IfModule>test`

- `module`
- `!module`

`module` `LoadModule` `module`

`module` `rewrite_module` `mod_rewrite.c`
`STANDARD20_MODULE_STUFF`

`<IfModule>`

`<IfModule>`



Include <i>file-path directory-path</i>
server config, virtual host, directory
(C)
core
Apache 2.0.41

Shell(fnmatch())

IncludeApache

[httpd](#)

()

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

ServerRoot

```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

Apache

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
```

- [apachect1](#)



KeepAlive

HTTP
KeepAlive On Off
KeepAlive On
server config, virtual host
(C)
core

Keep-Alive HTTP/1.0 HTTP/1.1 HTTP TCP HTML 50%
Apache 1.2 KeepAlive On

HTTP/1.0 HTTP/1.0 CGISSI HTTP/1.0
HTTP/1.1

- [MaxKeepAliveRequests](#)



KeepAliveTimeout

KeepAliveTimeout <i>seconds</i>
KeepAliveTimeout 5
server config, virtual host
(C)
core

Apache Timeout

KeepAliveTimeout



HTTP
<code><Limit method [method] ... > ... </Limit></code>
server config, virtual host, directory, .htaccess
All
(C)
core

`<Limit>`

`<Limit>`HTTP `<Limit>`POST, PUT, DELETE

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

GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH,
PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK GET
HEAD TRACE

`<LimitExcept>` `<Limit>` `<LimitExcept>`HTTP



LimitExcept

HTTP
<code><LimitExcept method [method] ... > ... </LimitExcept></code>
server config, virtual host, directory, .htaccess
All
(C)
core

`<LimitExcept></LimitExcept>` HTTP `<Limit>`

```
<LimitExcept POST GET>  
  Require valid-user  
</LimitExcept>
```



LimitInternalRecursion <i>number</i> [<i>number</i>]
LimitInternalRecursion 10
server config, virtual host
(C)
core
Apache 2.0.47

[ActionCGIApacheURI](#)

[mod_dirDirectoryIndex](#)

LimitInternalRecursion

number() *number* *number*

LimitInternalRecursion 5



LimitRequestBody

HTTP
LimitRequestBody <i>bytes</i>
LimitRequestBody 0
server config, virtual host, directory, .htaccess
All
(C)
core

bytes0)2147483647(2GB)

LimitRequestBody()HTTPCGI PUT

100K

LimitRequestBody 102400



LimitRequestFields

HTTP
LimitRequestFields <i>number</i>
LimitRequestFields 100
server config
(C)
core

Number0()32767 DEFAULT_LIMIT_REQUEST_FIELDS(
100)

LimitRequestFieldsHTTP20HTTP

LimitRequestFields 50



LimitRequestFields

LimitRequestFieldsize <i>bytes</i>
LimitRequestFieldsize 8190
server config
(C)
core

*bytes*HTTP

LimitRequestFieldSizeHTTPSPNEGO
12392

LimitRequestFieldSize 4094



LimitRequestLine

HTTP
LimitRequestLine <i>bytes</i>
LimitRequestLine 8190
server config
(C)
core

*bytes*HTTP

LimitRequestLineHTTPHTTPURI
LimitRequestLineURI GET

LimitRequestLine 4094



LimitXMLRequestBody

XML
LimitXMLRequestBody <i>bytes</i>
LimitXMLRequestBody 1000000
server config, virtual host, directory, .htaccess
All
(C)
core

XML" 0"

LimitXMLRequestBody 0



URL
<code><Location URL-path URL> ... </Location></code>
server config, virtual host
(C)
core

```
<Location>URL <Directory> </Location>
  <Location><Directory>, .htaccess, <Files>
```

```
<Location> <Location>URL
```

```
<Location>
```

```
<Location> <Directory><Files> <Location />
URL
```

```
()URL" /path/"URLURL
" scheme://servername/path"
```

```
URL" ?"" *"
```

```
" ~"
```

```
<Location ~ "/(extra|special)/data">
```

```
" /extra/data"" /special/data"URLApache1.3
<LocationMatch> <Location>
```

```
<Location><SetHandler foo.com
```

```
<Location /status>
  SetHandler server-status
  Order Deny,Allow
```

```
Deny from all
Allow from .foo.com
</Location>
```

"")

```
URL(" /home///foo" /home/foo")URL
  <LocationMatch><Location> <LocationMatch
^/abc>" /abc" //abc" <Location> <Location>
  <Location /abc/def>" /abc//def"
```

- [<Directory><Location><Files>](#)



LocationMatch

URL
<code><LocationMatch regex> ... </LocationMatch></code>
server config, virtual host
(C)
core

`<LocationMatch><Location>URL`

```
<LocationMatch "/(extra|special)/data">
```

`" /extra/data" /special/data"URL`

- [<Directory><Location><Files>](#)



LogLevel

LogLevel <i>level</i>
LogLevel warn
server config, virtual host
(C)
core

LogLevel(ErrorLog) *level*

Level		
emerg	()	"Child cannot open lock file. Exiting"
alert		"getpwuid: couldn't determine user name from uid"
crit		"socket: Failed to get a socket, exiting child"
error		"Premature end of script headers"
warn		"child process 1234 did not exit, sending another SIGHUP"
notice		"httpd: caught SIGBUS, attempting to dump core in ..."
info		"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug		"Opening config file ..."

LogLevel info noticewarn

crit

LogLevel notice

notice

syslog



MaxKeepAliveRequests

MaxKeepAliveRequests <i>number</i>
MaxKeepAliveRequests 100
server config, virtual host
(C)
core

MaxKeepAliveRequestsKeepAlive" 0"

MaxKeepAliveRequests 500



NAMEVIRTUALHOST

IP()
NameVirtualHost <i>addr[:port]</i>
server config
(C)
core

NameVirtualHost

addrIP

```
NameVirtualHost 111.22.33.44
```

NameVirtualHostIPPIP

```
"" _default_ NameVirtualHost IP(  
NameVirtualHostVirtualHost)
```

```
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 [+ -] <i>option</i> [[+ -] <i>option</i>] ...
Options All
server config, virtual host, directory, .htaccess
Options
(C)
core

Options

*option*None

All

MultiViews

ExecCGI

mod_cgiCGI

FollowSymLinks

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

Includes

mod_include

IncludesNOEXEC

```
" #exec cmd"" #exec cgi" ScriptAlias" #include  
virtual"CGI
```

Indexes

```
URL DirectoryIndex( index.html)  
mod_autoindex
```

MultiViews

`mod_negotiation` (MultiViews)

SymLinksIfOwnerMatch

uid

```
<Location>
```

```
Options ()( ) Options" +"" -"" +"
" _"
" +"" -"
```

```
<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>
```

```
<Directory /web/docs/spec>
  Options Includes
</Directory>
```

`Includes/web/docs/spec` `Options` +"" -"

```
<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>
```

```
<Directory /web/docs/spec>
  Options +Includes -Indexes
</Directory>
```

`FollowSymLinksIncludes/web/docs/spec`

-IncludesNOEXEC -Includes

All



Require

Require <i>entity-name</i> [<i>entity-name</i>] ...
directory, .htaccess
AuthConfig
(C)
core

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

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

Require valid-user

Require mod authz user, mod authz groupfile,
mod authz ldap, mod authz dbm, mod authz owner

RequireAuthNameAuthType AuthUserFileAuthGroupFile
()

```
AuthType Basic
AuthName "Restricted Resource"
AuthUserFile /web/users
AuthGroupFile /web/groups
Require group admin
```

Require<Limit>

RequireAllowDeny Satisfy

Satisfy mod_authz_host

```
<Directory /path/to/protected/>
  Require user david
</Directory>
<Directory /path/to/protected/unprotected>
  #
  Satisfy Any
  Allow from all
</Directory>
```

-
- Satisfy
- mod_authz_host



ApacheCPU
RLimitCPU <i>seconds</i> max [<i>seconds</i> max]
server config, virtual host, directory, .htaccess
All
(C)
core

" max" root

ApacheApacheCGISSIApache

CPU

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



Apache
<code>RLimitMEM bytes max [bytes max]</code>
server config, virtual host, directory, .htaccess
All
(C)
core

" max" root

ApacheApacheCGISSIApache

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



Apache
<code>RLimitNPROC number max [number max]</code>
server config, virtual host, directory, .htaccess
All
(C)
core

" max" root

ApacheApacheCGISSIApache

CGIwebuid error_log" cannot fork"

- [RLimitMEM](#)
- [RLimitCPU](#)



Satisfy

Satisfy Any All
Satisfy All
directory, .htaccess
AuthConfig
(C)
core
2.0.51<Limit><LimitExcept>

AllowRequire All Any / (All) Any

web

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

2.0.51 Satisfy<Limit><LimitExcept>

- Allow
- Require



Computer Security

CGI
ScriptInterpreterSource Registry Registry-Strict Script
ScriptInterpreterSource Script
server config, virtual host, directory, .htaccess
FileInfo
(C)
core
Win32 Registry-Strict Apache 2.0

ApacheCGI Script " #!"Win32

```
#!C:/Perl/bin/perl.exe
```

perlPATH

```
#!perl
```

ScriptInterpreterSource Registry (.pl)Windows
HKEY_CLASSES_ROOT Shell\ExecCGI\Command
Shell\Open\Command()Apache Script

```
ScriptInterpreterSource Registry ScriptAlias
Apache RegistryWindows .htmlE .htmlE
```

Registry-StrictRegistry Shell\ExecCGI\Command
ExecCGI



ServerAdmin <i>email-address</i> <i>URL</i>
server config, virtual host
(C)
core

ServerAdmin *httpdURLemail-address* mailto:Email
CGIURL

ServerAdmin *www-admin@foo.example.com*



ServerAlias

ServerAlias <i>hostname</i> [<i>hostname</i>] ...
virtual host
(C)
core

ServerAlias

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

- [Apache](#)



ServerName

ServerName <i>fully-qualified-domain-name[:port]</i>
server config, virtual host
(C)
core
2.01.3 Port

ServerNameURLweb simple.example.com DNS
www.example.com web

```
ServerName www.example.com:80
```

ServerNameIP ServerName ServerName

<VirtualHost>ServerName" Host:"

UseCanonicalNameUseCanonicalPhysicalPortURL(mod_dir)

- [DNSApache](#)
- [Apache](#)
- [UseCanonicalName](#)
- [UseCanonicalPhysicalPort](#)
- [NameVirtualHost](#)
- [ServerAlias](#)



ServerPath

URL
ServerPath <i>URL-path</i>
virtual host
(C)
core

ServerPath(legacy)URL

- [Apache](#)



ServerRoot

ServerRoot <i>directory-path</i>
ServerRoot /usr/local/apache
server config
(C)
core

ServerRoot conf/logs/ (IncludeLoadModule)

```
ServerRoot /home/httpd
```

- [httpd -d](#)
- ServerRoot



ServerSignature

ServerSignature On Off EMail
ServerSignature Off
server config, virtual host, directory, .htaccess
All
(C)
core

ServerSignature([mod_proxyftp](#) [mod_info](#))

Off (Apache1.2) On [ServerName](#) EMail
[ServerAdmin](#)"mailto:"

2.0.44 [ServerTokens](#)

- [ServerTokens](#)



" Server:"
ServerTokens Major Minor Min[imal] Prod[uctOnly] OS Full
ServerTokens Full
server config
(C)
core

" Server:"

ServerTokens Prod[uctOnly]

() Server: Apache

ServerTokens Major

() Server: Apache/2

ServerTokens Minor

() 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 <i>handler-name</i> None
server config, virtual host, directory, .htaccess
FileInfo
(C)
core
Apache2.0

```
.htaccess<Directory><Location> handler-name
      .htaccess
```

```
SetHandler imap-file
```

```
http://servername/status httpd.conf
```

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

None [SetHandler](#)

- [AddHandler](#)



SetInputFilter

POST
SetInputFilter <i>filter</i> [<i>;filter...</i>]
server config, virtual host, directory, .htaccess
FileInfo
(C)
core

SetInputFilterPOST([AddInputFilter](#))

(;)

-



SetOutputFilter

SetOutputFilter <i>filter</i> [<i>;filter...</i>]
server config, virtual host, directory, .htaccess
FileInfo
(C)
core

SetOutputFilter([AddOutputFilter](#))

/www/data/SSI

```
<Directory /www/data/>  
  SetOutputFilter INCLUDES  
</Directory>
```

(;)

-

-

TimeOut

TimeOut <i>seconds</i>
TimeOut 300
server config
(C)
core

TimeOutApache

1. GET
2. POSTPUTTCP
3. TCPACK

1.21200300



TRACE

TRACE
TraceEnable <i>[on off extended]</i>
TraceEnable on
server config
(C)
core
Apache 1.3.34, 2.0.55

```
mod_proxyTRACE( TraceEnable on)RFC2616TRACE  
TraceEnable off mod_proxy" 405"()
```

```
" TraceEnable extended"()64k( Transfer-  
Encoding: chunked HTTP8k)64k
```



UseCanonicalName On Off DNS
UseCanonicalName Off
server config, virtual host, directory
(C)
core

Apache URL(URL) UseCanonicalName On ServerName
 URL SERVER_NAMECGISERVER_PORT

UseCanonicalName Off ()ApacheURL CGI
 SERVER_NAMESERVER_PORT

www http://www/splatURL Apache
 http://www.domain.com/splat/ www
 www.domain.com([FAQ](#)) UseCanonicalName Off Apache
 http://www/splat/

UseCanonicalName DNS IP" Host:"ApacheIPDNS
 URL

CGISERVER_NAMECGI SERVER_NAMEURL

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



UseCanonicalPhysicalPort On Off
UseCanonicalPhysicalPort Off
server config, virtual host, directory
(C)
core
Apache 2.2.0

Apache URL(URL) UseCanonicalPhysicalPort On
 Apache UseCanonicalName(physical p
 UseCanonicalPhysicalPort Off Apache

UseCanonicalName On

- Servername
-
-

UseCanonicalName Off | DNS

- "Host:"
-
- Servername
-

UseCanonicalPhysicalPort Off

- UseCanonicalName
- ServerName
- Listen



IP
<code><VirtualHost addr[:port] [addr[:port]] ...> ...</code> <code></VirtualHost></code>
server config
(C)
core

`<VirtualHost></VirtualHost>`

Addr

- IP
- IP
- " *"" NameVirtualHost *"IP
- " _default_"IP

`<VirtualHost>`

```
<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>
```

IPv6IPv6

```
<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>
```

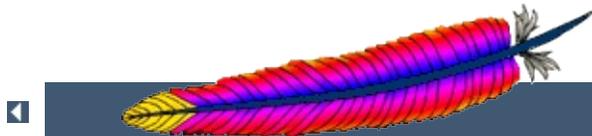
IPIPIP(ifconfig alias)

```
<VirtualHost>Apache ListenApache
```

```
IP" _default_"IP" _default_"IP"()
    NameVirtualHostIP"" _default_"
```

```
" :port" Listen" :*"(" _default_"
```

- [Apache](#)
- [DNSApache](#)
- [Apache](#)
- [<Directory><Location><Files>](#)



◀ **Apache HTTP Server 2.2**

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

| | | 2006122 |

Apache MPM

(MPM)

MPM



AcceptMutex

Apache()(socket)
AcceptMutex Default <i>method</i>
AcceptMutex Default
server config
MPM
<u>prefork, worker</u>

AcceptMutex()2.0

Default

flock

flock(2)([LockFile](#))

fcntl

fcntl(2)([LockFile](#))

posixsem

(2.0)POSIXsegfault

pthread

(1.3)POSIXPOSIXSolaris2.5

sysvsem

(1.3)SysVSysVApache([ipcs\(\) man page](#))
APIuidCGI(CGI)

[LogLevel](#)[debug](#) [AcceptMutex](#)[ErrorLog](#)

```
pthread AcceptCnt1Solaris(Apache)
pthread_mutexattr_setrobust_np() pthread
```



CoreDumpDirectory

Apache
CoreDumpDirectory <i>directory</i>
server config
MPM
<u>beos, mpm_winnt, prefork, worker</u>

Apache [ServerRoot](#)

Linux

ApacherootLinux ApacheApache2.0.46
CoreDumpDirectoryLinux2.4



EnableExceptionHook

EnableExceptionHook On Off
EnableExceptionHook Off
server config
MPM
<u>prefork, worker</u>
Apache 2.0.49

--enable-exception-hook (hook)

(mod_whatkilledusmod_backtrace)Jeff Trawick

[EnableExceptionHook site](#)



GracefulShutdownTimeout <i>seconds</i>
GracefulShutdownTimeout 0
server config
MPM
<u>prefork, worker, event</u>
Apache 2.2

GracefulShutdownTimeout""

"0"



Group

Apache
Group <i>unix-group</i>
Group #-1
server config
MPM
<u>beos, mpmt_os2, prefork, worker</u>
Apache2.0

GroupApacheApache

root

Unix-group

"#"(GID)

Group www-group

Apache nobody

Group(User)root

<VirtualHost> suexecSuexecUserGroup

Groupbeosmpmt_os2MPM



IP
<code>Listen [IP-address:]portnumber [protocol]</code>
server config
MPM
<u>beos</u> , <u>mpm_netware</u> , <u>mpm_winnt</u> , <u>mpmt_os2</u> , <u>prefork</u> , <u>worker</u> , <u>event</u>
Apache2.0 <i>protocol</i> 2.1.5

`ListenApacheIPApacheIP` `Listen`

`Listen`

`Listen/`

`808000`

```
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
```

`protocol443` `https` `http` [AcceptFilter](#)

`protocol8443` `https`

```
Listen 192.170.2.1:8443 https
```

```
Listen" Address already in use"
```

- [DNS](#)

-



ListenBacklog

(pending connection)
ListenBacklog <i>backlog</i>
ListenBacklog 511
server config
MPM
<u>beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker</u>

(pending connection)TCP SYN

listen(2)

()



LOCKING

LockFile <i>filename</i>
LockFile logs/accept.lock
server config
MPM
<u>prefork, worker</u>

LockFile AcceptMutex fcntl flock Apache
PID

logs NFS

(/var/tmp)

- AcceptMutex



MAXCLIENTS

MaxClients <i>number</i>
server config
MPM
<u>beos, prefork, worker</u>

MaxClients MaxClients ListenBacklog

MPM(prefork) MaxClients 256 ServerLimit

MPM(beosworker) MaxClients beos50MPM
16(ServerLimit)25(ThreadsPerChild) MaxClients16
ServerLimit



MEMORY

free()(KB)
MaxMemFree <i>KBytes</i>
MaxMemFree 0
server config
MPM
<u>beos, mpm_netware, prefork, worker, mpm_winnt</u>

MaxMemFreefree()(KB)"0"



MaxRequestsPerChild

MaxRequestsPerChild <i>number</i>
MaxRequestsPerChild 10000
server config
MPM
<u>mpm_netware, mpm_winnt, mpmt_os2, prefork, worker</u>

```
MaxRequestsPerChild MaxRequestsPerChild  
MaxRequestsPerChild" 0"
```

```
mpm_netwarempm_winnt" 0"
```

MaxRequestsPerChild

- ()
-

KeepAlive



MaxSpareThreads

MaxSpareThreads <i>number</i>
server config
MPM
<u>beos</u> , <u>mpm_netware</u> , <u>mpmt_os2</u> , <u>worker</u>

MPM

worker" 250"MPM

mpm_netware" 100"MPMPM

beosmpmt_os2mpm_netware beos" 50" mpmt_os2" 10"

MaxSpareThreadsApache

- mpm_netwareMinSpareThreads
- workerMinSpareThreadsThreadsPerChild

- MinSpareThreads
- StartServers



MinSpareThreads

MinSpareThreads <i>number</i>
server config
MPM
<u>beos</u> , <u>mpm_netware</u> , <u>mpmt_os2</u> , <u>worker</u>

MPM

worker" 75"MPM

mpm_netware" 10"MPMMPM

beosmpmt_os2mpm_netware beos" 1" mpmt_os2" 5"

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



TABLE

()PID
PidFile <i>filename</i>
PidFile logs/httpd.pid
server config
MPM
<u>beos, mpm_winnt, mpmt_os2, prefork, worker</u>

PidFile()PID ServerRoot

```
PidFile /var/run/apache.pid
```

ErrorLogTransferLog"SIGHUP"(kill -1) PidFile
PID

PidFile

```
Apache2    apachectl
```



RECEIVEBUFFER

TCP()
ReceiveBufferSize <i>bytes</i>
ReceiveBufferSize 0
server config
MPM
<u>beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker</u>

TCP()(100ms)

" 0"



(coordination data)
ScoreBoardFile <i>file-path</i>
ScoreBoardFile logs/apache_status
server config
MPM
<u>beos, mpm_winnt, prefork, worker</u>

Apache(scoreboard)Apache(scoreboard)
Apache

```
ScoreBoardFile /var/run/apache_status
```

(scoreboard)

ScoreBoardFileRAM disk

- [Apache](#)



Configuration

TCP()
SendBufferSize <i>bytes</i>
SendBufferSize 0
server config
MPM
<u>beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker</u>

TCP()(100ms)

" 0"



ServerLimit <i>number</i>
server config
MPM
<u>prefork, worker</u>

preforkMPM MaxClients workerMPM ThreadLimit
MaxClients MaxClients

ServerLimit ServerLimitMaxClientsApache

preforkMPM MaxClients256 MaxClients

workerMPM MaxClientsThreadsPerChild16 MaxClients
ThreadsPerChild

```
Apache" ServerLimit 20000"( preforkMPM
" ServerLimit 200000")
```

- [Apache](#)



STARTSERVERS

StartServers <i>number</i>
server config
MPM
<u>mpmt_os2</u> , <u>prefork</u> , <u>worker</u>

StartServers

MPM worker" 3" prefork" 5" mpmt_os2" 2"



StartThreads

StartThreads <i>number</i>
server config
MPM
<u>beos</u> , <u>mpm_netware</u>

mpm_netware " 50"

beos " 10"



THREADS PER CHILD

ThreadLimit <i>number</i>
server config
MPM
<u>mpm_winnt</u> , <u>worker</u>
2.0.41 <u>mpm_winnt</u>

ThreadsPerChild ThreadsPerChild

 ThreadLimitThreadsPerChild ThreadLimit
ThreadsPerChildApache ThreadsPerChild

mpm_winntThreadLimit1920MPM64

```
Apache" ThreadLimit 20000"( mpm_winnt" ThreadLimit  
15000")
```



ThreadsPerChild

ThreadsPerChild <i>number</i>
server config
MPM
<u>mpm_winnt</u> , <u>worker</u>

mpm_winntMPM

workerMPM

mpm_winntThreadsPerChild64MPM25



()
ThreadStackSize <i>size</i>
NetWare65536
server config
MPM
<u>mpm_netware, mpm_winnt, worker</u>
Apache 2.1

ThreadStackSize()

- (HP-UX)Apache ThreadStackSize
- ThreadStackSize ThreadStackSize



User <i>unix-userid</i>
User #-1
server config
MPM
<u>prefork, worker</u>
2.0

User root root root root *Unix-userid*

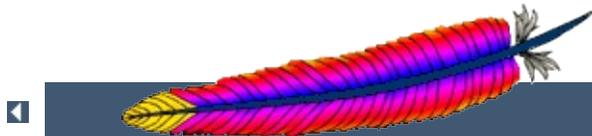
"#"

Apache nobody

User(Group)root

<VirtualHost> suexecSuexecUserGroup

Userbeosmpmt_os2MPM



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache MPM beos

This Multi-Processing Module is optimized for BeOS.
MPM
mpm_beos_module
beos.c

This Multi-Processing Module (MPM) is the default for BeOS. It uses a single control process which creates threads to handle requests.



MaxRequestsPerThread

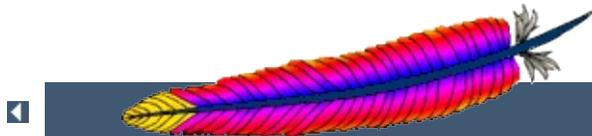
Limit on the number of requests that an individual thread will handle during its life
MaxRequestsPerThread <i>number</i>
MaxRequestsPerThread 0
server config
MPM
beos

`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.

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.



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

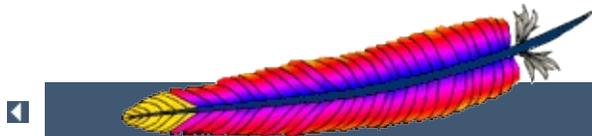
Apache MPM event

An experimental variant of the standard worker MPM
MPM
mpm_event_module
event.c

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

To use the [event](#) MPM, add `--with-mpm=event` 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.



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache MPM netware

Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare
MPM
mpm_netware_module
mpm_netware.c

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.

[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.

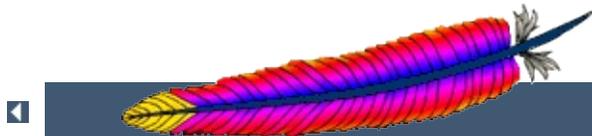


MAXTHREADS

Set the maximum number of worker threads
MaxThreads <i>number</i>
MaxThreads 2048
server config
MPM
mpm_netware

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
```



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

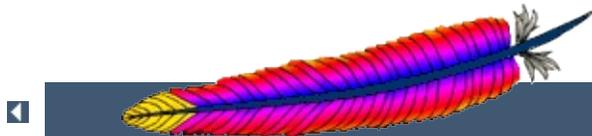
Apache MPM os2

Hybrid multi-process, multi-threaded MPM for OS/2
MPM
mpm_mpmt_os2_module
mpmt_os2.c

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](#)[MaxSpareThreads](#).



◀ Apache HTTP Server 2.2

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

| | | 2006121 |

Apache MPM prefork

MPM
MPM
mpm_prefork_module
prefork.c

(MPM)webApache 1.3MPM

MPM MaxClients



()Apache (spare)

StartServers, MinSpareServers, MaxSpareServers,
MaxClientsApache256 MaxClients MaxClients

Unix root80Apache UserGroup

MaxRequestsPerChild



MaxSpareServers

MaxSpareServers <i>number</i>
MaxSpareServers 10
server config
MPM
prefork

MaxSpareServers

MaxSpareServers

MinSpareServersApache"

MinSpareServers+1"

- MinSpareServers
- StartServers



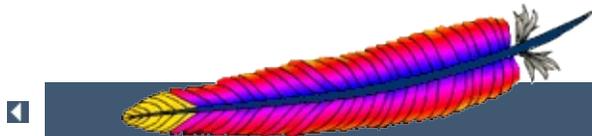
minspareServers

MinSpareServers <i>number</i>
MinSpareServers 5
server config
MPM
prefork

MinSpareServers

MinSpareServers Apache

- [MaxSpareServers](#)
- [StartServers](#)



Apache HTTP Server 2.2

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

| | | 2006121 |

Apache MPM winnt

Windows NTMPM
MPM
mpm_winnt_module
mpm_winnt.c

(MPM)Windows NT





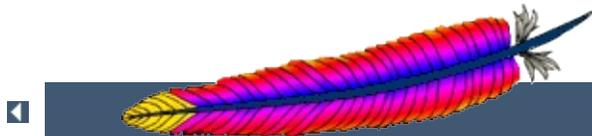
accept()AcceptEx()
Win32DisableAcceptEx
AcceptEx()
server config
MPM
mpm_winnt
Apache 2.0.49

AcceptEx() WinSock2 API BSDaccept() APIWindows
AcceptEx()

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

AcceptEx()





Apache HTTP Server 2.2

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

| | | 2006121 |

Apache MPM worker

MPM
mpm_worker_module
worker.c

(MPM)MPMPM

MPM ThreadsPerChild MaxClients



() ThreadsPerChild

Apache(spare) StartServers MinSpareThreads
MaxSpareThreads MaxClients MaxClients
ThreadsPerChild

() ServerLimit MaxClientsThreadsPerChild
ThreadLimit ThreadsPerChild workerMPM

"" MaxClients

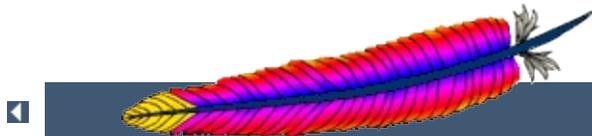
- MaxRequestsPerChild"0"
- MaxSpareThreadsMaxClients

workerMPM

```
ServerLimit 16
StartServers 2
MaxClients 150
MinSpareThreads 25
MaxSpareThreads 75
ThreadsPerChild 25
```

Unix80 rootApache UserGroupApache
suexec CGI

MaxRequestsPerChild



◀ **Apache HTTP Server 2.2**

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

| | | 2006122 |

Apache mod_actions

CGI
(B)
actions_module
mod_actions.c

ActionMIMECGI ScriptCGICGI



CGI
Action <i>action-type cgi-script</i> [virtual]
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_actions
virtual Apache 2.1

action-type *cgi-script* *cgi-script*URL [ScriptAliasAddHandler](#)
 CGI *action-type* [MIME](#)PATH_INFO PATH_TRANSLATED URL
 REDIRECT_HANDLER

```
# MIME
Action image/gif /cgi-bin/images.cgi

#
AddHandler my-file-type .xyz
Action my-file-type /cgi-bin/program.cgi
```

MIME" image/gif"CGI /cgi-bin/images.cgi

" .xyz"CGI /cgi-bin/program.cgi

virtual [Action](#)

```
<Location /news>
  SetHandler news-handler
  Action news-handler /cgi-bin/news.cgi virtual
</Location>
```

- AddHandler



Script

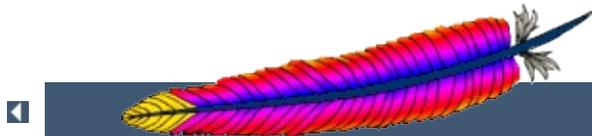
CGI
<code>Script method cgi-script</code>
server config, virtual host, directory
(B)
mod_actions

```
methodcgi-script  cgi-scriptURL  ScriptAliasAddHandlerCGI  
PATH_INFOPATH_TRANSLATEDURL
```

```
Script PUT  Script put
```

```
ScriptCGI      GET("foo.html?hi")
```

```
# <ISINDEX>  
Script GET /cgi-bin/search  
  
# A CGI PUT  
Script PUT /~bob/put.cgi
```



◀ **Apache HTTP Server 2.2**

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

| | | 2006123 |

Apache mod_alias

URL
(B)
alias_module
mod_alias.c

URL AliasScriptAliasURL DocumentRoot
ScriptAliasCGI

RedirectURL

mod_aliasURLURL

mod_rewrite



(context) (context)(<VirtualHost>)

RedirectRedirectMatch

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



Alias

URL
Alias <i>URL -path file-path directory-path</i>
server config, virtual host
(B)
mod_alias

Alias DocumentRoot(%) *url-path* URL *directory-path*

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

"http://myserver/image/foo.gif"/ftp/pub/image/foo.gif"

"http://myserver/imagefoo.gif"

AliasM:

```
url-path"/"/" Alias /icons/  
/usr/local/apache/icons/" /icons"
```

<Directory><Directory>(<Location>)

DocumentRootAlias

```
Alias /image /ftp/pub/image  
<Directory /ftp/pub/image>  
    Order allow,deny  
    Allow from all  
</Directory>
```



AliasMatch

URL
AliasMatch <i>regex file-path directory-path</i>
server config, virtual host
(B)
mod_alias

Alias URL-path" /icons"

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



Redirect

URL
Redirect [<i>status</i>] <i>URL-path</i> <i>URL</i>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_alias

URLURLURL

URL-path(%)*"|"*()

URL(%)*"|"*()*URL*

URLURL-path

URL-path *URL*

```
Redirect /service http://foo2.example.com/service
```

"http://example.com/service/foo.txt"

"http://foo2.example.com/service/foo.txt"

"http://example.com/servicefoo.txt"

[Redi](#)

```
AliasScriptAlias
```

status"(HTTP status 302)

*status*HTTP

permanent

(301)

temp

(302)

seeother

""(303)

gone

""(410)

URL

status300-399 *URL*Apache(http_protocol.c
send_error_response)

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



RedirectMatch

URL
RedirectMatch [<i>status</i>] <i>regex</i> <i>URL</i>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_alias

Redirect *regex*URL-pathGIFJPEG

```
RedirectMatch (.*)\.gif$  
http://www.anotherserver.com$1.jpg
```



Redirect permanent

URL
RedirectPermanent <i>URL-path</i> <i>URL</i>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_alias

(status 301)" Redirect permanent"



RedirectTemp

URL
RedirectTemp <i>URL-path</i> URL
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_alias

(status 302)" Redirect temp"



ScriptAlias

URLCGI
<code>ScriptAlias URL-path file-path directory-path</code>
server config, virtual host
(B)
mod_alias

`ScriptAlias` `Alias` `cgi-scriptCGI` `URL-path(%)URL`

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

`http://myserver/cgi-bin/foo/web/cgi-bin/foo`

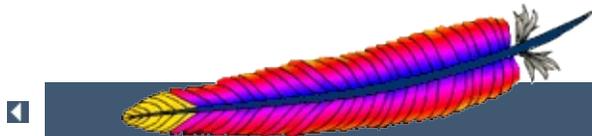


ScriptAliasMatch

URLCGI
ScriptAliasMatch <i>regex file-path directory-path</i>
server config, virtual host
(B)
mod_alias

ScriptAlias *regex*URL-path /cgi-bin

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



Apache HTTP Server 2.2

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

| | | 2006123 |

Apache mod_asis

HTTP
(B)
asis_module
mod_asis.c

send-as-isApacheHTTP(headers)

HTTPcgi-scriptnph script

MIME httpd/send-as-is



```
send-as-is
```

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

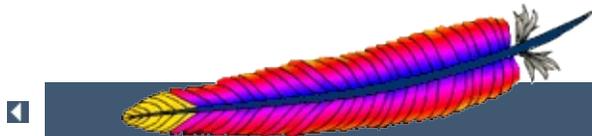
```
" .asis"ApacheHTTP"Status:"3HTTP
```

```
Status: 301 Now where did I leave that URL  
Location: http://xyz.abc.com/foo/bar.html  
Content-type: text/html
```

```
<html>  
<head>  
<title>Lame excuses'R'us</title>  
</head>  
<body>  
<h1>Fred's exceptionally wonderful page has moved  
to  
<a  
href="http://xyz.abc.com/foo/bar.html">Joe's</a>  
site.  
</h1>  
</body>  
</html>
```

```
" Date:"" Server:"      " Last-Modified:"
```

||||



Apache HTTP Server 2.2

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

| | | 2006123 |

Apache mod_auth_basic

(B)
auth_basic_module
mod_auth_basic.c
Apache 2.1

HTTP mod_auth_digestHTTP(
mod_authn_file)(mod_authz_user)



()
AuthBasicAuthoritative On Off
AuthBasicAuthoritative On
directory, .htaccess
AuthConfig
(B)
mod_auth_basic

AuthBasicProvider AuthBasicAuthoritative
 Off userID **userIDrule()** (non-provider-based)
mod_auth_basicAuthBasicProvider

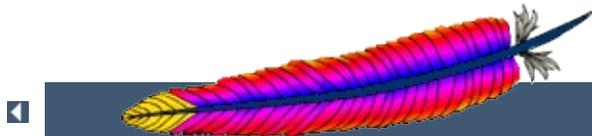


<code>()(Provider)</code>
<code>AuthBasicProvider provider-name [provider-name]</code>
<code>...</code>
<code>AuthBasicProvider file</code>
<code>directory, .htaccess</code>
<code>AuthConfig</code>
<code>(B)</code>
<code>mod_auth_basic</code>

`AuthBasicProvider()`(Provider) `file`[mod_authn_file](#)(
DSO)

```
<Location /secure>
  AuthType basic
  AuthBasicProvider dbm
  AuthDBMType SDBM
  AuthDBMUserFile /www/etc/dbmpasswd
  Require valid-user
</Location>
```

(Provider) [mod_authn_dbm](#), [mod_authn_file](#),
[mod_authn_dbd](#), [mod_authnz_ldap](#)



Apache HTTP Server 2.2

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

| | | 2006123 |

Apache mod_auth_digest

MD5()
(X)
auth_digest_module
mod_auth_digest.c

HTTP



MD5" AuthType Digest" [AuthDigestProvider](#)
" AuthType Basic" [AuthBasicProvider](#)
[AuthDigestDomainURI](#)

[htdigest\(\)](#)

```
<Location /private/>
  AuthType Digest
  AuthName "private area"
  AuthDigestDomain /private/
  http://mirror.my.dom/private2/

  AuthDigestProvider file
  AuthUserFile /web/auth/.digest_pw
  Require valid-user
</Location>
```

20049 [Amaya](#), [Konqueror](#), [MS Internet Explorer 6](#)(" [MS Internet Explorer 6](#) _____"), [Mozilla](#), [Netscape 7](#), [Opera](#), [Safari](#)
[lynx](#)



Internet Explorer 6 GETRFC

POSTGET

2.0.51Apache AuthDigestEnableQueryStringHack
(workaround) AuthDigestEnableQueryStringHackApache
Internet Explorer 6 bug URI

MSIE6

```
BrowserMatch "MSIE"  
AuthDigestEnableQueryStringHack=On
```

[BrowserMatch](#)



AuthDigestAlgorithm

AuthDigestAlgorithm MD5 MD5-sess
AuthDigestAlgorithm MD5
directory, .htaccess
AuthConfig
(X)
mod_auth_digest

AuthDigestAlgorithm

MD5-sess



AuthDigestDomain

URI
AuthDigestDomain <i>URI</i> [<i>URI</i>] ...
directory, .htaccess
AuthConfig
(X)
mod_auth_digest

AuthDigestDomainURI(/)URIURI""URI/URI
URI()URI

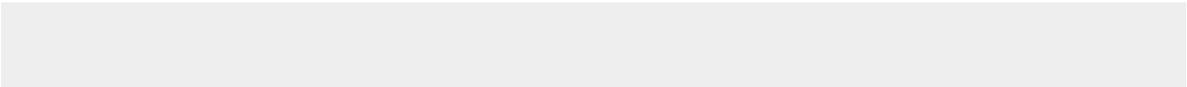
URI AuthDigestNcCheck"On"

URI



AuthDigestNcCheck

Enables or disables checking of the nonce-count sent by the server
AuthDigestNcCheck On Off
AuthDigestNcCheck Off
server config
(X)
mod_auth_digest



Determines how the nonce is generated

AuthDigestNonceFormat *format*

directory, .htaccess

AuthConfig

(X)

mod_auth_digest



AuthDigestNonceLifetime

nonce()
AuthDigestNonceLifetime <i>seconds</i>
AuthDigestNonceLifetime 300
directory, .htaccess
AuthConfig
(X)
mod_auth_digest

```
AuthDigestNonceLifetime nonce() nonce()  
" stale=true"401() seconds"0"nonce()()30120(  
10)
```



AuthDigestProvider

<code>()(Provider)</code>
<code>AuthDigestProvider provider-name [provider-name]</code>
<code>...</code>
<code>AuthDigestProvider file</code>
<code>directory, .htaccess</code>
<code>AuthConfig</code>
<code>(X)</code>
<code>mod_auth_digest</code>

`AuthDigestProvider()`(Provider) [file](#)[mod_authn_file](#)
(DSO)

(Provider) [mod_authn_dbm](#)[mod_authn_file](#)



AuthDigestQop

AuthDigestQop none auth auth-int [auth auth-int]
AuthDigestQop auth
directory, .htaccess
AuthConfig
(X)
mod_auth_digest

AuthDigestQop(*quality-of-protection*)auth(/) auth-int(
MD5) noneRFC-2069() authauth-int non

auth-int



AuthDigestShmemSize

AuthDigestShmemSize <i>size</i>
AuthDigestShmemSize 1000
server config
(X)
mod_auth_digest

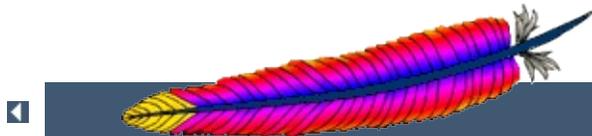
AuthDigestShmemSize

AuthDigestShmemSize

" 0"Apache

size" K"" M"KBMB

```
AuthDigestShmemSize 1048576
AuthDigestShmemSize 1024K
AuthDigestShmemSize 1M
```



Apache HTTP Server 2.2

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

| | | 2006123 |

Apache mod_auth_alias

(E)
auth_alias_module
mod_auth_alias.c
Apache 2.1

[AuthBasicProviderAuthDigestProvider](#)



ldap()ldap()ldap

```
LoadModule authn_alias_module
modules/mod_authn_alias.so

<AuthnProviderAlias ldap ldap-alias1>
  AuthLDAPBindDN cn=youruser,o=ctx
  AuthLDAPBindPassword yourpassword
  AuthLDAPURL ldap://ldap.host/o=ctx
</AuthnProviderAlias>

<AuthnProviderAlias ldap ldap-other-alias>
  AuthLDAPBindDN cn=yourotheruser,o=dev
  AuthLDAPBindPassword yourotherpassword
  AuthLDAPURL ldap://other.ldap.host/o=dev?cn
</AuthnProviderAlias>

Alias /secure /webpages/secure
<Directory /webpages/secure>
  Order deny,allow
  Allow from all

  AuthBasicProvider ldap-other-alias ldap-alias1

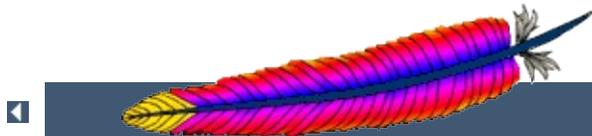
  AuthType Basic
  AuthName LDAP_Protected_Place
  AuthzLDAPAuthoritative off
  require valid-user
</Directory>
```



AuthnProviderAlias

<code><AuthnProviderAlias <i>baseProvider Alias</i>> ...</code> <code></AuthnProviderAlias></code>
server config, virtual host
(E)
mod_auth_alias

`<AuthnProviderAlias></AuthnProviderAlias>`
`AuthBasicProviderAuthDigestProvider`



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_auth_anon

(E)
authn_anon_module
mod_auth_anon.c
Apache 2.1

This module provides authentication front-ends such as [mod_auth_basic](#) to authenticate users 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.

When using [mod_auth_basic](#), this module is invoked via the [AuthBasicProvider](#) directive with the anon value.



The example below is combined with "normal" htpasswd-file based authentication and allows users in additionally 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, *i.e.* 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](#))

```
<Directory /foo>
  AuthName "Use 'anonymous' & Email address for
  guest entry"
  AuthType Basic
  AuthBasicProvider file anon
  AuthUserFile /path/to/your/.htpasswd

  Anonymous_NoUserID off
  Anonymous_MustGiveEmail on
  Anonymous_VerifyEmail on
  Anonymous_LogEmail on
  Anonymous anonymous guest www test welcome

  Order Deny,Allow
  Allow from all

  Require valid-user
</Directory>
```



Anonymous

Specifies userIDs that are allowed access without password verification
<code>Anonymous user [user] ...</code>
directory, .htaccess
AuthConfig
(E)
mod_authn_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**.

It's strongly recommended that the magic username 'anonymous' is always one of the allowed userIDs.

```
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".

As of Apache 2.1 it is possible to specify the userID as "*". That allows *any* supplied userID to be accepted.



Anonymous_LogEmail

Sets whether the password entered will be logged in the error log
Anonymous_LogEmail On Off
Anonymous_LogEmail On
directory, .htaccess
AuthConfig
(E)
mod_auth_anon

When set On, the default, the 'password' entered (which hopefully contains a sensible email address) is logged in the error log.



Anonymous_MustGiveEmail

Specifies whether blank passwords are allowed
Anonymous_MustGiveEmail On Off
Anonymous_MustGiveEmail On
directory, .htaccess
AuthConfig
(E)
mod_authn_anon

Specifies whether the user must specify an email address as the password. This prohibits blank passwords.



Sets whether the userID field may be empty
Anonymous_NoUserID On Off
Anonymous_NoUserID Off
directory, .htaccess
AuthConfig
(E)
mod_authn_anon

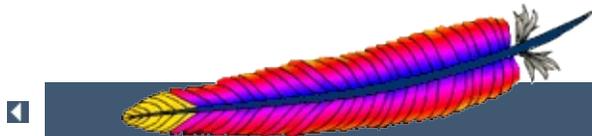
When set On, users can leave the userID (and perhaps the password field) empty. This can be very convenient for MS-Explorer users who can just hit return or click directly on the OK button; which seems a natural reaction.



Anonymous_VerifyEmail

Sets whether to check the password field for a correctly formatted email address
Anonymous_VerifyEmail On Off
Anonymous_VerifyEmail Off
directory, .htaccess
AuthConfig
(E)
mod_authn_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](#)).



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_auth_dbd

SQL
(E)
authn_dbd_module
mod_auth_dbd.c
Apache 2.1

This module provides authentication front-ends such as [mod_auth_digest](#) and [mod_auth_basic](#) to authenticate users by looking up users in SQL tables. Similar functionality is provided by, for example, [mod_auth_file](#).

This module relies on [mod_dbd](#) to specify the backend database driver and connection parameters, and manage the database connections.

When using [mod_auth_basic](#) or [mod_auth_digest](#), this module is invoked via the [AuthBasicProviderAuthDigestProvider](#) with the dbd value.



This simple example shows use of this module in the context of the Authentication and DBD frameworks.

```
#Database Management

#Use the PostgreSQL driver
DBDriver pgsq1

#Connection string: database name and login credentials
DBDParams "dbname=htpasswd user=apache pass=xxxxxx"

#Parameters for Connection Pool Management
DBDMin 1
DBDKeep 2
DBDMax 10
DBDExptime 60

#Authentication Section
<Directory /usr/www/myhost/private>

    #mod_auth configuration for authn_dbd
    AuthType Basic
    AuthName "My Server"
    AuthBasicProvider dbd

    #authz configuration
    Require valid-user

    #SQL query to verify a user
    #(note: DBD drivers recognise both stdio-like %s ;
    AuthDBDUserPWQuery "select password from authn where
</Directory>
```



AuthDBUserPWQuery

SQL query to look up a password for a user
AuthDBUserPWQuery <i>query</i>
directory
AuthConfig
(E)
mod_authn_dbd

AuthDBUserPWQuery specifies an SQL query to look up a password for a specified user. The query must take a single string (typically SQL varchar) argument (username), and return a single value (encrypted password).

```
AuthDBUserPWQuery "SELECT password FROM authn  
WHERE username = %s"
```

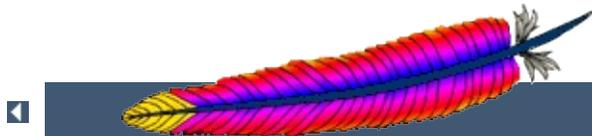


AuthDBDUserRealmQuery

SQL query to look up a password hash for a user and realm.
AuthDBDUserRealmQuery <i>query</i>
directory
AuthConfig
(E)
mod_authn_dbd

AuthDBDUserRealmPWQuery specifies an SQL query to look up a password for a specified user and realm. The query must take two string (typically SQL varchar) arguments (username and realm), and return a single value (encrypted password).

```
AuthDBDUserRealmPWQuery "SELECT password FROM  
authn WHERE username = %s AND realm = %s"
```



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_auth_dbm

DBM
(E)
authn_dbm_module
mod_auth_dbm.c
Apache 2.1

This module provides authentication front-ends such as [mod_auth_digest](#) and [mod_auth_basic](#) to authenticate users by looking up users in *dbm* password files. Similar functionality is provided by [mod_auth_file](#).

When using [mod_auth_basic](#) or [mod_auth_digest](#), this module is invoked via the [AuthBasicProviderAuthDigestProvider](#) with the *dbm* value.



AuthDBMType

Sets the type of database file that is used to store passwords
AuthDBMType default SDBM GDBM NDBM DB
AuthDBMType default
directory, .htaccess
AuthConfig
(E)
mod_authn_dbm

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.



Sets the name of a database file containing the list of users and passwords for authentication
AuthDBMUserFile <i>file-path</i>
directory, .htaccess
AuthConfig
(E)
mod_authn_dbm

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.

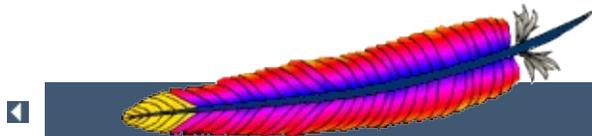
Make sure that the **AuthDBMUserFile** is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients will be able to download the **AuthDBMUserFile**.

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.





Apache HTTP Server 2.2

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

| | | 2006124 |

Apache mod_auth_default

(B)
authn_default_module
mod_auth_default.c
Apache 2.1

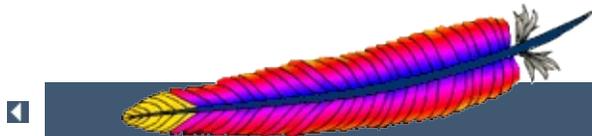
(fallback)([mod_auth_basic](#))



AuthDefaultAuthoritative On Off
AuthDefaultAuthoritative On
directory, .htaccess
AuthConfig
(B)
mod_auth_default

AuthDefaultAuthoritative Off (modules.c)

mod_auth_default AuthDefaultAuthoritative
(On)



◀ **Apache HTTP Server 2.2**

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

| | | 2006124 |

Apache mod_auth_file

(B)
authn_file_module
mod_auth_file.c
Apache 2.1

(mod_auth_digest mod_auth_basic) mod_authn_dbm
mod_auth_basic mod_auth_digest AuthBasicProvider
AuthDigestProvider file



/
AuthUserFile <i>file-path</i>
directory, .htaccess
AuthConfig
(B)
mod_authn_file

AuthUserFile/ File-path() ServerRoot

mod_authn_file

(" src/support") htpasswdHTTP

usernameFilename

```
htpasswd -c Filename username
```

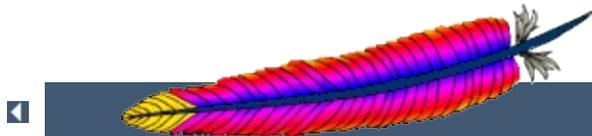
Filenameusername2

```
htpasswd Filename username2
```

AuthDBMUserFile

HTTPhtpasswd htdigest

AuthUserFileWEB



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_authnz_ldap

LDAP
(E)
authnz_ldap_module
mod_authnz_ldap.c
Apache 2.1

This module provides authentication front-ends such as [mod_auth_basic](#) to authenticate users through an ldap directory.

[mod_authnz_ldap](#) supports the following features:

- Known to support the [OpenLDAP SDK](#) (both 1.x and 2.x), [Novell LDAP SDK](#) and the [iPlanet \(Netscape\) SDK](#).
- Complex authorization policies can be implemented by representing the policy with LDAP filters.
- Uses extensive caching of LDAP operations via [mod_ldap](#).
- Support for LDAP over SSL (requires the Netscape SDK) or TLS (requires the OpenLDAP 2.x SDK or Novell LDAP SDK).

When using [mod_auth_basic](#), this module is invoked via the [AuthBasicProvider](#) directive with the `ldap` value.



-
- [Operation](#)
 - [The Authentication Phase](#)
 - [The Authorization Phase](#)

 - [The require Directives](#)
 - [require valid-user](#)
 - [require ldap-user](#)
 - [require ldap-group](#)
 - [require ldap-dn](#)
 - [require ldap-attribute](#)
 - [require ldap-filter](#)

 - [Examples](#)
 - [Using TLS](#)
 - [Using SSL](#)
 - [Using Microsoft FrontPage with `mod_authnz_ldap`](#)
 - [How It Works](#)
 - [Caveats](#)



There are two phases in granting access to a user. The first phase is authentication, in which the [mod_authnz_ldap](#) authentication provider verifies that the user's credentials are valid. This is also called the *search/bind* phase. The second phase is authorization, in which [mod_authnz_ldap](#) determines if the authenticated user is allowed access to the resource in question. This is also known as the *compare* phase.

[mod_authnz_ldap](#) registers both an `authn_ldap` authentication provider and an `authz_ldap` authorization handler. The `authn_ldap` authentication provider can be enabled through the [AuthBasicProvider](#) directive using the `ldap` value. The `authz_ldap` handler extends the [Require](#) directive's authorization types by adding `ldap-user`, `ldap-dnldap-group` values.

The Authentication Phase

During the authentication phase, [mod_authnz_ldap](#) searches for an entry in the directory that matches the username that the HTTP client passes. If a single unique match is found, then [mod_authnz_ldap](#) attempts to bind to the directory server using the DN of the entry plus the password provided by the HTTP client. Because it does a search, then a bind, it is often referred to as the *search/bind* phase. Here are the steps taken during the *search/bind* phase.

1. Generate a search filter by combining the attribute and filter provided in the [AuthLDAPURL](#) directive with the username passed by the HTTP client.
2. Search the directory using the generated filter. If the search does not return exactly one entry, deny or decline access.
3. Fetch the distinguished name of the entry retrieved from the search and attempt to bind to the LDAP server using the DN and the password passed by the HTTP client. If the bind is

unsuccessful, deny or decline access.

The following directives are used during the search/bind phase

AuthLDAPURL	Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to use.
AuthLDAPBindDN	An optional DN to bind with during the search phase.
AuthLDAPBindPassword	An optional password to bind with during the search phase.

The Authorization Phase

During the authorization phase, [mod_authnz_ldap](#) attempts to determine if the user is authorized to access the resource. Many of these checks require [mod_authnz_ldap](#) to do a compare operation on the LDAP server. This is why this phase is often referred to as the compare phase. [mod_authnz_ldap](#) accepts the following [Require](#) directives to determine if the credentials are acceptable:

- Grant access if there is a [require_ldap-user](#) directive, and the username in the directive matches the username passed by the client.
- Grant access if there is a [require_ldap-dn](#) directive, and the DN in the directive matches the DN fetched from the LDAP directory.
- Grant access if there is a [require_ldap-group](#) directive, and the DN fetched from the LDAP directory (or the username passed by the client) occurs in the LDAP group.
- Grant access if there is a [require_ldap-attribute](#) directive, and the attribute fetched from the LDAP directory matches the given value.
- Grant access if there is a [require_ldap-filter](#) directive, and

the search filter successfully finds a single user object that matches the dn of the authenticated user.

- otherwise, deny or decline access

Other **Require** values may also be used which may require loading additional authorization modules.

- Grant access if there is a **require valid-user** directive. (requires **mod_authz_user**)
- Grant access if there is a **require group** directive, and **mod_authz_groupfile** has been loaded with the **AuthGroupFile** directive set.
- others...

mod_authnz_ldap uses the following directives during the compare phase:

<u>AuthLDAPURL</u>	The attribute specified in the URL is used in compare operations for the require ldap-user operation.
<u>AuthLDAPCompareDNOnServer</u>	Determines the behavior of the require ldap-dn directive.
<u>AuthLDAPGroupAttribute</u>	Determines the attribute to use for comparisons in the require ldap-group directive.
<u>AuthLDAPGroupAttributeIsDN</u>	Specifies whether to use the user DN or the username when doing comparisons for the require ldap-group directive.



Apache's [Require](#) directives are used during the authorization phase to ensure that a user is allowed to access a resource.

`mod_authnz_ldap` extends the authorization types with `ldap-user`, `ldap-dn`, `ldap-group`, `ldap-attribute` and `ldap-filter`. Other authorization types may also be used but may require that additional authorization modules be loaded.

require valid-user

If this directive exists, `mod_authnz_ldap` grants access to any user that has successfully authenticated during the search/bind phase.

Requires that `mod_authz_user` be loaded and that the [AuthzLDAPAuthoritative](#) directive be set to off.

require ldap-user

`require ldap-user` directive specifies what usernames can access the resource. Once `mod_authnz_ldap` has retrieved a unique DN from the directory, it does an LDAP compare operation using the username specified in the `require ldap-user` to see if that username is part of the just-fetched LDAP entry. Multiple users can be granted access by putting multiple usernames on the line, separated with spaces. If a username has a space in it, then it must be surrounded with double quotes. Multiple users can also be granted access by using multiple `require ldap-user` directives, with one user per line. For example, with a [AuthLDAPURL](#) of `ldap://ldap/o=Airius?cn` (i.e., `cn` is used for searches), the following `require` directives could be used to restrict access:

```
require ldap-user "Barbara Jenson"  
require ldap-user "Fred User"  
require ldap-user "Joe Manager"
```

Because of the way that `mod_authnz_ldap` handles this directive,

Barbara Jenson could sign on as *Barbara Jenson*, *Babs Jenson* or any other cn that she has in her LDAP entry. Only the single `require ldap-user` line is needed to support all values of the attribute in the user's entry.

If the `uid` attribute was used instead of the `cn` attribute in the URL above, the above three lines could be condensed to

```
require ldap-user bjenson fuser jmanager
```

require ldap-group

This directive specifies an LDAP group whose members are allowed access. It takes the distinguished name of the LDAP group. Note: Do not surround the group name with quotes. For example, assume that the following entry existed in the LDAP directory:

```
dn: cn=Administrators, o=Airius
objectClass: groupOfUniqueNames
uniqueMember: cn=Barbara Jenson, o=Airius
uniqueMember: cn=Fred User, o=Airius
```

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

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

Behavior of this directive is modified by the [AuthLDAPGroupAttributeAuthLDAPGroupAttributeIsDN](#) directives.

require ldap-dn

`require ldap-dn` directive allows the administrator to grant access based on distinguished names. It specifies a DN that must match for

access to be granted. If the distinguished name that was retrieved from the directory server matches the distinguished name in the `require ldap-dn`, then authorization is granted. Note: do not surround the distinguished name with quotes.

The following directive would grant access to a specific DN:

```
require ldap-dn cn=Barbara Jenson, o=Airius
```

Behavior of this directive is modified by the [AuthLDAPCompareDNOnServer](#) directive.

require ldap-attribute

`require ldap-attribute` directive allows the administrator to grant access based on attributes of the authenticated user in the LDAP directory. If the attribute in the directory matches the value given in the configuration, access is granted.

The following directive would grant access to anyone with the attribute `employeeType = active`

```
require ldap-attribute employeeType=active
```

Multiple attribute/value pairs can be specified on the same line separated by spaces or they can be specified in multiple `require ldap-attribute` directives. The effect of listing multiple attribute/values pairs is an OR operation. Access will be granted if any of the listed attribute values match the value of the corresponding attribute in the user object. If the value of the attribute contains a space, only the value must be within double quotes.

The following directive would grant access to anyone with the city attribute equal to "San Jose" or status equal to "Active"

```
require ldap-attribute city="San Jose"  
status=active
```

require ldap-filter

`require ldap-filter` directive allows the administrator to grant access based on a complex LDAP search filter. If the dn returned by the filter search matches the authenticated user dn, access is granted.

The following directive would grant access to anyone having a cell phone and is in the marketing department

```
require ldap-filter &(cell=*)  
(department=marketing)
```

The difference between the `require ldap-filter` directive and the `require ldap-attribute` directive is that `ldap-filter` performs a search operation on the LDAP directory using the specified search filter rather than a simple attribute comparison. If a simple attribute comparison is all that is required, the comparison operation performed by `ldap-attribute` will be faster than the search operation used by `ldap-filter` especially within a large directory.



- Grant access to anyone who exists in the LDAP directory, using their UID for searches.

```
AuthLDAPURL  
ldap://ldap1.airius.com:389/ou=People,  
o=Airius?uid?sub?(objectClass=*)  
require valid-user
```

- The next example is the same as above; but with the fields that have useful defaults omitted. Also, note the use of a redundant LDAP server.

```
AuthLDAPURL ldap://ldap1.airius.com  
ldap2.airius.com/ou=People, o=Airius  
require valid-user
```

- The next example is similar to the previous one, but it uses the common name instead of the UID. Note that this could be problematical if multiple people in the directory share the same cn, because a search on cn **must** return exactly one entry. That's why this approach is not recommended: it's a better idea to choose an attribute that is guaranteed unique in your directory, such as uid.

```
AuthLDAPURL ldap://ldap.airius.com/ou=People,  
o=Airius?cn  
require valid-user
```

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

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?  
uid
```

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

- The next example assumes that everyone at Airius who carries an alphanumeric pager will have an LDAP attribute of `qpagePagerID`. The example will grant access only to people (authenticated via their UID) who have alphanumeric pagers:

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?  
uid??(qpagePagerID=*)  
require valid-user
```

- The next example demonstrates the power of using filters to accomplish complicated administrative requirements. Without filters, it would have been necessary to create a new LDAP group and ensure that the group's members remain synchronized with the pager users. This becomes trivial with filters. The goal is to grant access to anyone who has a pager, plus grant access to Joe Manager, who doesn't have a pager, but does need to access the same resource:

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?  
uid??(|(qpagePagerID=*)(uid=jmanager))  
require valid-user
```

This last may look confusing at first, so it helps to evaluate what the search filter will look like based on who connects, as shown below. If Fred User connects as `fuser`, the filter would look like

```
(&(|(qpagePagerID=*)(uid=jmanager))  
(uid=fuser))
```

The above search will only succeed if *fuser* has a pager. When Joe Manager connects as *jmanager*, the filter looks like

```
(&( |(qpagePagerID=*)(uid=jmanager))  
(uid=jmanager))
```

The above search will succeed whether *jmanager* has a pager or not.



Using TLS

To use TLS, see the [mod_ldap](#) directives [LDAPTrustedClientCert](#), [LDAPTrustedGlobalCert](#) and [LDAPTrustedMode](#).

An optional second parameter can be added to the [AuthLDAPURL](#) to override the default connection type set by [LDAPTrustedMode](#). This will allow the connection established by an *ldap://* Url to be upgraded to a secure connection on the same port.



To use SSL, see the [mod_ldap](#) directives [LDAPTrustedClientCert](#), [LDAPTrustedGlobalCert](#) and [LDAPTrustedMode](#).

To specify a secure LDAP server, use *ldaps://* in the [AuthLDAPURL](#) directive, instead of *ldap://*.



Normally, FrontPage uses FrontPage-web-specific user/group files (i.e., the `mod_authn_file` `mod_authz_groupfile` modules) to handle all authentication. Unfortunately, it is not possible to just change to LDAP authentication by adding the proper directives, because it will break the *Permissions* forms in the FrontPage client, which attempt to modify the standard text-based authorization files.

Once a FrontPage web has been created, adding LDAP authentication to it is a matter of adding the following directives to every `.htaccess` file that gets created in the web

```
AuthLDAPURL           "the url"  
AuthzLDAPAuthoritative off  
AuthGroupFile mygroupfile  
require group mygroupfile
```

`AuthzLDAPAuthoritative` must be off to allow `mod_authnz_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. The `require valid-user` directive will succeed for any user who is valid as far as LDAP is concerned. This means that anybody who has an entry in the LDAP directory is considered a valid user, whereas FrontPage considers only those people in the local user file to be valid. By substituting the `ldap-group` with `group file` authorization, Apache is allowed to consult the local user file (which is managed by FrontPage) - instead of LDAP - when handling authorizing the user.

Once directives have been added as specified above, FrontPage

users will be able to perform all management operations from the FrontPage client.

Caveats

- When choosing the LDAP URL, the attribute to use for authentication should be something that will also be valid for putting into a [mod_authn_file](#) user file. The user ID is ideal for this.
- When adding users via FrontPage, FrontPage administrators should choose usernames that already exist in the LDAP directory (for obvious reasons). Also, the password that the administrator enters into the form is ignored, since Apache will actually be authenticating against the password in the LDAP database, and not against the password in the local user file. This could cause confusion for web administrators.
- Apache must be compiled with [mod_auth_basic](#), [mod_authn_file](#), [mod_authz_groupfile](#) in order to use FrontPage support. This is because Apache will still use the [mod_authz_groupfile](#) group file for determine the extent of a user's access to the FrontPage web.
- The directives must be put in the .htaccess files. Attempting to put them inside `<Location><Directory>` directives won't work. This is because [mod_authnz_ldap](#) has to be able to grab the [AuthGroupFile](#) directive that is found in FrontPage .htaccess files so that it knows where to look for the valid user list. If the [mod_authnz_ldap](#) directives aren't in the same .htaccess file as the FrontPage directives, then the hack won't work, because [mod_authnz_ldap](#) will never get a chance to process the .htaccess file, and won't be able to find the FrontPage-managed user file.



Optional DN to use in binding to the LDAP server
AuthLDAPBindDN <i>distinguished-name</i>
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

An optional DN used to bind to the server when searching for entries. If not provided, [mod_authnz_ldap](#) will use an anonymous bind.



Password used in conjunction with the bind DN
AuthLDAPBindPassword <i>password</i>
directory, .htaccess
AuthConfig
(E)
mod_authnz_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](#) [AuthLDAPBindPassword](#) if you absolutely need them to search the directory.



Language to charset conversion configuration file
<code>AuthLDAPCharsetConfig</code> <i>file-path</i>
server config
(E)
<code>mod_authnz_ldap</code>

`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.



Use the LDAP server to compare the DN's
AuthLDAPCompareDNOnServer on off
AuthLDAPCompareDNOnServer on
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

When set, [mod_authnz_ldap](#) will use the LDAP server to compare the DN's. This is the only foolproof way to compare DN's. [mod_authnz_ldap](#) will search the directory for the DN specified with the [require dn](#) directive, then, retrieve the DN and compare it with the DN retrieved from the user entry. If this directive is not set, [mod_authnz_ldap](#) simply does a string comparison. It is possible to get false negatives with this approach, but it is much faster. Note the [mod_ldap](#) cache can speed up DN comparison in most situations.



When will the module de-reference aliases
AuthLDAPDereferenceAliases never searching finding always
AuthLDAPDereferenceAliases Always
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

This directive specifies when [mod_authnz_ldap](#) will de-reference aliases during LDAP operations. The default is `always`.



AuthLDAPGroupAttribute

LDAP attributes used to check for group membership
AuthLDAPGroupAttribute <i>attribute</i>
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

This directive specifies which LDAP attributes are used to check for group membership. Multiple attributes can be used by specifying this directive multiple times. If not specified, then [mod_authnz_ldap](#) uses the memberuniquemember attributes.



AuthLDAPGroupAttributeIsDN

Use the DN of the client username when checking for group membership
AuthLDAPGroupAttributeIsDN on off
AuthLDAPGroupAttributeIsDN on
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

When set on, this directive says to use the distinguished name of the client username when checking for group membership. Otherwise, the username will be used. For example, assume that the client sent the username bjenson, which corresponds to the LDAP DN cn=Babs Jenson, o=Airius. If this directive is set, [mod_authnz_ldap](#) will check if the group has cn=Babs Jenson, o=Airius as a member. If this directive is not set, then [mod_authnz_ldap](#) will check if the group has bjenson as a member.



AuthLDAPRemoteUserIsDN

Use the DN of the client username to set the REMOTE_USER environment variable
AuthLDAPRemoteUserIsDN on off
AuthLDAPRemoteUserIsDN off
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

If this directive is set to on, the value of the REMOTE_USER environment variable will be set to the full distinguished name of the authenticated user, rather than just the username that was passed by the client. It is turned off by default.



URL specifying the LDAP search parameters
AuthLDAPUrl <i>url</i> [<i>NONE</i> <i>SSL</i> <i>TLS</i> <i>STARTTLS</i>]
directory, .htaccess
AuthConfig
(E)
mod_authnz_ldap

An RFC 2255 URL which specifies the LDAP search parameters to use. The syntax of the URL is

```
ldap://host:port/basedn?attribute?scope?filter
```

ldap

For regular ldap, use the string ldap. For secure LDAP, use ldaps instead. Secure LDAP is only available if Apache was linked to an LDAP library with SSL support.

host:port

The name/port of the ldap server (defaults to localhost : 389 for ldap, and localhost : 636 for ldaps). To specify multiple, redundant LDAP servers, just list all servers, separated by spaces. `mod_authnz_ldap` will try connecting to each server in turn, until it makes a successful connection.

Once a connection has been made to a server, that connection remains active for the life of the `httpd` process, or until the LDAP server goes down.

If the LDAP server goes down and breaks an existing connection, `mod_authnz_ldap` will attempt to re-connect, starting with the primary server, and trying each redundant server in turn. Note that this is different than a true round-robin search.

basedn

The DN of the branch of the directory where all searches should start from. At the very least, this must be the top of your directory tree, but could also specify a subtree in the directory.

attribute

The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute will be used, no matter how many are provided. If no attributes are provided, the default is to use `uid`. It's a good idea to choose an attribute that will be unique across all entries in the subtree you will be using.

scope

The scope of the search. Can be either `onesub`. 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))`.

An optional parameter can be added to allow the LDAP Url to override

the connection type. This parameter can be one of the following:

NONE

Establish an unsecure connection on the default LDAP port. This is the same as `ldap://` on port 389.

SSL

Establish a secure connection on the default secure LDAP port. This is the same as `ldaps://`

TLS | STARTTLS

Establish an upgraded secure connection on the default LDAP port. This connection will be initiated on port 389 by default and then upgraded to a secure connection on the same port.

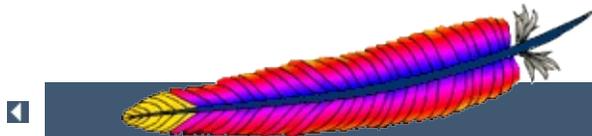
See above for examples of [AuthLDAPURL](#) URLs.



AuthzLDAPAuthoritative

Prevent other authentication modules from authenticating the user if this one fails
AuthzLDAPAuthoritative on off
AuthzLDAPAuthoritative on
directory, .htaccess
AuthConfig
(E)
mod_authnz_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).



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_auth_dbm

DBM
(E)
authz_dbm_module
mod_auth_dbm.c
Apache 2.1

This module provides authorization capabilities so that authenticated users can be allowed or denied access to portions of the web site by group membership. Similar functionality is provided by [mod_auth_groupfile](#).



AuthDBMGroupFile

Sets the name of the database file containing the list of user groups for authorization
<code>AuthDBMGroupFile</code> <i>file-path</i>
directory, .htaccess
AuthConfig
(E)
mod_authz_dbm

`AuthDBMGroupFile` directive sets the name of a DBM file containing the list of user groups for user authorization. *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.

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

```
Encrypted 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 authorization module. This is what www.telescope.org uses for its combined password and group database.



Sets whether authorization will be passed on to lower level modules
AuthzDBMAuthoritative On Off
AuthzDBMAuthoritative On
directory, .htaccess
AuthConfig
(E)
mod_authz_dbm

Setting the `AuthzDBMAuthoritative` directive explicitly to `Off` allows group authorization to be passed on to lower level modules (as defined in the `modules.c` file) if there is no group found for the the supplied `userID`. If there are any groups specified, the usual 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 `AuthBasicAuthoritative` setting.

A common use for this is in conjunction with one of the auth providers; such as `mod_authn_dbm` or `mod_authn_file`. 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 group will result in an Authentication Required reply. Not setting it thus keeps the system secure and forces an NCSA compliant behaviour.

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.

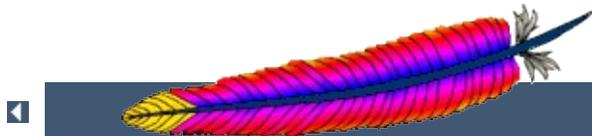


AuthzDBMType

Sets the type of database file that is used to store list of user groups
AuthzDBMType default SDBM GDBM NDBM DB
AuthzDBMType default
directory, .htaccess
AuthConfig
(E)
mod_authz_dbm

Sets the type of database file that is used to store the list of user groups. 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 group files is configured to use the same type of database.



Apache HTTP Server 2.2

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

| | | 2006124 |

Apache mod_authz_default

(B)
authz_default_module
mod_authz_default.c
Apache 2.1

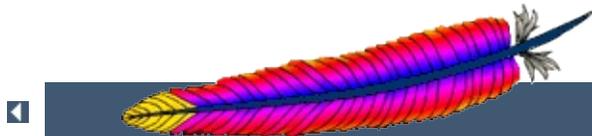
(fallback)([mod_authz_usermod_authz_groupfile](#))



AuthzDefaultAuthoritative On Off
AuthzDefaultAuthoritative On
directory, .htaccess
AuthConfig
(B)
mod_authz_default

AuthzDefaultAuthoritative Off(modules.c)

mod_authz_default AuthzDefaultAuthoritative
(On)



◀ **Apache HTTP Server 2.2**

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

| | | 2006124 |

Apache mod_auth_groupfile

(B)
authz_groupfile_module
mod_auth_groupfile.c
Apache 2.1

[mod_auth_dbm](#)



AuthGroupFile

AuthGroupFile <i>file-path</i>
directory, .htaccess
AuthConfig
(B)
mod_authz_groupfile

AuthGroupFile *File-path* ServerRoot

```
mygroup: bob joe anne
```

AuthDBMGroupFile

```
AuthGroupFileWEB
```



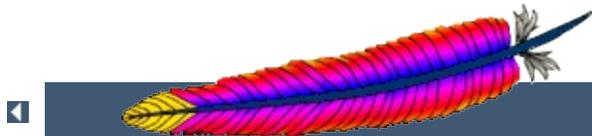
AuthzGroupFileAuthoritative

AuthzGroupFileAuthoritative On Off
AuthzGroupFileAuthoritative On
directory, .htaccess
AuthConfig
(B)
mod_authz_groupfile

AuthzGroupFileAuthoritative Off userID()
(modules.c)

NCSA

.htaccess .htpasswd



◀ Apache HTTP Server 2.2

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

| | | 2006124 |

Apache mod_authz_host

IP
(B)
authz_host_module
mod_authz_host.c
Apache 2.1

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



ALLOW

<code>Allow from all host env=env-variable [host env=env-variable] ...</code>
directory, .htaccess
Limit
(B)
mod_authz_host

AllowIP IP

```
" from"" Allow from all" DenyOrder host  
  
()
```

```
Allow from apache.org  
Allow from .net example.edu
```

```
foo.apache.orgfooapache.org Apache  
HostnameLookupsIPDNSIP
```

IP

```
Allow from 10.1.2.3  
Allow from 192.168.1.104 192.168.1.205
```

IP

IP

```
Allow from 10.1
Allow from 10 172.20 192.168.2
```

IP13

/

```
Allow from 10.1.0.0/255.255.0.0
```

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

/nnn(CIDR specification)

```
Allow from 10.1.0.0/16
```

nnn

IPv6IPv6

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

Allow" Allow from env=*env-variable*" *env-variable*
mod_setenvif User-Agent() RefererHTTP

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

```
    Allow from env=let_me_in  
</Directory>
```

KnockKnock/2.0



Deny

Deny from all <i>host</i> env= <i>env-variable</i> [<i>host</i> env= <i>env-variable</i>] ...
directory, .htaccess
Limit
(B)
mod_authz_host

IP [DenyAllow](#)



<u>AllowDeny</u>
Order <i>ordering</i>
Order Deny,Allow
directory, .htaccess
Limit
(B)
mod_authz_host

OrderAllowDeny *Ordering*

Deny, Allow

DenyAllow DenyAllow

Allow, Deny

AllowDeny AllowDeny

Mutual-failure

AllowDeny" Order Allow,Deny"

AllowDeny

apache.org

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

apache.orgfoo.apache.orgapache.org

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

Order" Deny,Allow""

Allow from

apache.org"" Deny from foo.apache.org" apache.org

AllowDeny Order

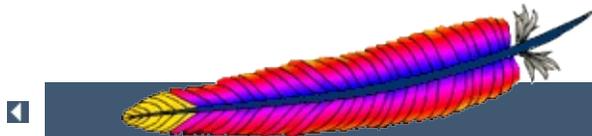
```
<Directory /www>  
  Order Allow,Deny  
</Directory>
```

/www

Order <Location>AllowDeny<Directory>.htaccess

AllowDeny Order

|||



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_authz_owner

(E)
authz_owner_module
mod_authz_owner.c
Apache 2.1

This module authorizes access to files by comparing the userid used for HTTP authentication (the web userid) with the file-system owner or group of the requested file. The supplied username and password must be already properly verified by an authentication module, such as [mod_auth_basic](#) or [mod_auth_digest](#).

[mod_authz_owner](#) recognizes two arguments for the [Require](#) directive, `file-owner` or `file-group`, as follows:

file-owner

The supplied web-username must match the system's name for the owner of the file being requested. That is, if the operating system says the requested file is owned by `jones`, then the username used to access it through the web must be `jones` as well.

file-group

The name of the system group that owns the file must be present in a group database, which is provided, for example, by [mod_authz_groupfile](#) or [mod_authz_dbm](#), and the web-username must be a member of that group. For example, if the operating system says the requested file is owned by `(system) group accounts`, the `accounts` must appear in the group database and the web-username used in the request must be a member of that group.

If [mod_auth_owner](#) is used in order to authorize a resource that is not actually present in the filesystem (*i.e.* a virtual resource), it will deny the access.

Particularly it will never authorize [content negotiated "MultiViews"](#) resources.



Require file-owner

Consider a multi-user system running the Apache Web server, with each user having his or her own files in `~/public_html/private`. Assuming that there is a single [AuthDBMUserFile](#) database that lists all of their web-username, and that these usernames match the system's usernames that actually own the files on the server, then the following stanza would allow only the user himself access to his own files. User `jones` would not be allowed to access files in `/home/smith/public_html/private` unless they were owned by `jones` instead of `smith`.

```
<Directory /home/*/public_html/private>
  AuthType Basic
  AuthName MyPrivateFiles
  AuthBasicProvider dbm
  AuthDBMUserFile /usr/local/apache2/etc/.htdbm-
  all
  Satisfy All
  Require file-owner
</Directory>
```

Require file-group

Consider a system similar to the one described above, but with some users that share their project files in `~/public_html/project-foo`. The files are owned by the system group `foo` and there is a single [AuthDBMGroupFile](#) database that contains all of the web-username and their group membership, *i.e.* they must be at least member of a group named `foo`. So if `jones` and `smith` are both member of the group `foo`, then both will be authorized to access the `project-foo` directories of each other.

```
<Directory /home/*/public_html/project-foo>
  AuthType Basic
  AuthName "Project Foo Files"
  AuthBasicProvider dbm

  # combined user/group database
  AuthDBMUserFile /usr/local/apache2/etc/.htdbm-
all
  AuthDBMGroupFile /usr/local/apache2/etc/.htdbm-
all

  Satisfy All
  Require file-group
</Directory>
```



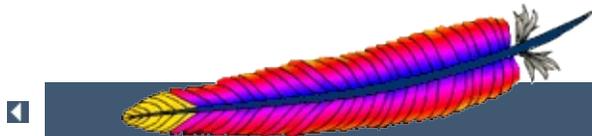
Sets whether authorization will be passed on to lower level modules
AuthzOwnerAuthoritative On Off
AuthzOwnerAuthoritative On
directory, .htaccess
AuthConfig
(E)
mod_authz_owner

Setting the `AuthzOwnerAuthoritative` directive explicitly to `Off` allows for user authorization to be passed on to lower level modules (as defined in the `modules.c` files) if:

- in the case of `file-owner` the file-system owner does not match the supplied web-username or could not be determined, or
- in the case of `file-group` the file-system group does not contain the supplied web-username or could not be determined.

Note that setting the value to `Off` also allows the combination of `file-ownerfile-group`, so access will be allowed if either one or the other (or both) match.

By default, control is not passed on and an authorization failure will result in an "Authentication Required" reply. Not setting it to `Off` thus keeps the system secure and forces an NCSA compliant behaviour.



Apache HTTP Server 2.2

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

| | | 2006124 |

Apache mod_authz_user

(B)
authz_user_module
mod_authz_user.c
Apache 2.1

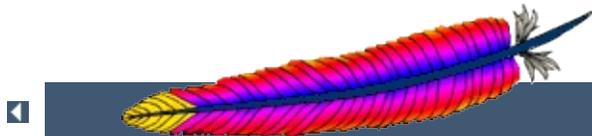
`mod_authz_user()` Require user require
valid-user



AuthzUserAuthoritative On Off
AuthzUserAuthoritative On
directory, .htaccess
AuthConfig
(B)
mod_authz_user

```
AuthzUserAuthoritative Off userID() ( modules.c
)
```

NCSA



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_autoindex

"ls""dir"
(B)
autoindex_module
mod_autoindex.c

The index of a directory can come from one of two sources:

- A file written by the user, typically called `index.html`. The [DirectoryIndex](#) directive sets the name of this file. This is controlled by [mod_dir](#).
- Otherwise, a listing generated by the server. The other directives control the format of this listing. The [AddIcon](#), [AddIconByEncodingAddIconByType](#) are used to set a list of icons to display for various file types; for each file listed, the first icon listed that matches the file is displayed. These are controlled by [mod_autoindex](#).

The two functions are separated so that you can completely remove (or replace) automatic index generation should you want to.

Automatic index generation is enabled with using `Options +Indexes`. See the [Options](#) directive for more details.

If the [FancyIndexing](#) option is given with the [IndexOptions](#) directive, the column headers are links that control the order of the display. If you select a header link, the listing will be regenerated, sorted by the values in that column. Selecting the same header repeatedly toggles between ascending and descending order. These column header links are suppressed with [IndexOptions](#)

directive's SuppressColumnSorting option.

Note that when the display is sorted by "Size", it's the *actual* size of the files that's used, not the displayed value - so a 1010-byte file will always be displayed before a 1011-byte file (if in ascending order) even though they both are shown as "1K".



Autoindex Request Query Arguments

Apache 2.0.23 reorganized the Query Arguments for Column Sorting, and introduced an entire group of new query options. To effectively eliminate all client control over the output, the [IndexOptions IgnoreClient](#) option was introduced.

The column sorting headers themselves are self-referencing hyperlinks that add the sort query options shown below. Any option below may be added to any request for the directory resource.

- C=N sorts the directory by file name
- C=M sorts the directory by last-modified date, then file name
- C=S sorts the directory by size, then file name
- C=D sorts the directory by description, then file name

- O=A sorts the listing in Ascending Order
- O=D sorts the listing in Descending Order

- F=0 formats the listing as a simple list (not FancyIndexed)
- F=1 formats the listing as a FancyIndexed list
- F=2 formats the listing as an HTMLTable FancyIndexed list

- V=0 disables version sorting
- V=1 enables version sorting

- P=*pattern* lists only files matching the given *pattern*

Note that the 'P'attern query argument is tested *after* the usual [IndexIgnore](#) directives are processed, and all file names are still subjected to the same criteria as any other autoindex listing. The Query Arguments parser in [mod_autoindex](#) will stop abruptly when an unrecognized option is encountered. The Query Arguments must be well formed, according to the table above.

The simple example below, which can be clipped and saved in a

header.html file, illustrates these query options. Note that the unknown "X" argument, for the submit button, is listed last to assure the arguments are all parsed before mod_autoindex encounters the X=Go input.

```
<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

Alternate text to display for a file, instead of an icon selected by filename
AddAlt <i>string file [file] ...</i>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

AddAlt provides the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *File* is a file extension, partial filename, wild-card expression or full filename for files to describe. If *String* contains any whitespace, you have to enclose it in quotes (" "). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

```
AddAlt "PDF file" *.pdf
AddAlt Compressed *.gz *.zip *.Z
```



Adding MIME-encoding

Alternate text to display for a file instead of an icon selected by MIME-encoding
<code>AddAltByEncoding string MIME-encoding [MIME-encoding] ...</code>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

`AddAltByEncoding` provides the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *MIME-encoding* is a valid content-encoding, such as *x-compress*. If *String* contains any whitespace, you have to enclose it in quotes (" "). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

```
AddAltByEncoding gzip x-gzip
```



AddAltByType

Alternate text to display for a file, instead of an icon selected by MIME content-type
AddAltByType <i>string</i> <i>MIME-type</i> [<i>MIME-type</i>] ...
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

AddAltByType sets the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *MIME-type* is a valid content-type, such as `text/html`. If *String* contains any whitespace, you have to enclose it in quotes (`" "`). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

```
AddAltByType 'plain text' text/plain
```



AddDescription

Description to display for a file
AddDescription <i>string file [file] ...</i>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

This sets the description to display for a file, for [FancyIndexing](#). *File* is a file extension, partial filename, wild-card expression or full filename for files to describe. *String* is enclosed in double quotes (").

```
AddDescription "The planet Mars"  
/web/pics/mars.gif
```

The typical, default description field is 23 bytes wide. 6 more bytes are added by the [IndexOptions SuppressIcon](#) option, 7 bytes are added by the [IndexOptions SuppressSize](#) option, and 19 bytes are added by the [IndexOptions SuppressLastModified](#) option. Therefore, the widest default the description column is ever assigned is 55 bytes.

See the [DescriptionWidth](#) [IndexOptions](#) keyword for details on overriding the size of this column, or allowing descriptions of unlimited length.

Caution

Descriptive text defined with `AddDescription` may contain HTML markup, such as tags and character entities. If the width of the description column should happen to truncate a tagged element (such as cutting off the end of a bolded phrase), the results may

affect the rest of the directory listing.



ADDICON

Icon to display for a file selected by name
AddIcon <i>icon name</i> [<i>name</i>] ...
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

This sets the icon to display next to a file ending in *name* for [FancyIndexing](#). *Icon* is either a (%-escaped) relative URL to the icon, or of the format (*alttext*, *url*) where *alttext* is the text tag given for an icon for non-graphical browsers.

Name is either ^^DIRECTORY^^ for directories, ^^BLANKICON^^ for blank lines (to format the list correctly), a file extension, a wildcard expression, a partial filename or a complete filename.

```
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm
AddIcon /icons/dir.xbm ^^DIRECTORY^^
AddIcon /icons/backup.xbm *~
```

[AddIconByType](#) should be used in preference to [AddIcon](#), when possible.



AddIconByEncoding

Icon to display next to files selected by MIME content-encoding
AddIconByEncoding <i>icon MIME-encoding [MIME-encoding] ...</i>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

This sets the icon to display next to files with [FancyIndexing](#). *Icon* is either a (%-escaped) relative URL to the icon, or of the format (*alttext*, *url*) where *alttext* is the text tag given for an icon for non-graphical browsers.

MIME-encoding is a wildcard expression matching required the content-encoding.

```
AddIconByEncoding /icons/compress.xbm x-compress
```



AddIconByType

Icon to display next to files selected by MIME content-type
AddIconByType <i>icon MIME-type</i> [<i>MIME-type</i>] ...
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

This sets the icon to display next to files of type *MIME-type* for [FancyIndexing](#). *Icon* is either a (%-escaped) relative URL to the icon, or of the format (*alttext*, *url*) where *alttext* is the text tag given for an icon for non-graphical browsers.

MIME-type is a wildcard expression matching required the mime types.

```
AddIconByType (IMG,/icons/image.xbm) image/*
```



Icon to display for files when no specific icon is configured
<code>DefaultIcon <i>url-path</i></code>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

`DefaultIcon` directive sets the icon to display for files when no specific icon is known, for [FancyIndexing](#). *Url-path* is a (%-escaped) relative URL to the icon.

```
DefaultIcon /icon/unknown.xbm
```



HeaderName

Name of the file that will be inserted at the top of the index listing
HeaderName <i>filename</i>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

HeaderName directive sets the name of the file that will be inserted at the top of the index listing. *Filename* is the name of the file to include.

```
HeaderName HEADER.html
```

Both HeaderName and **ReadmeName** now treat *Filename* as a URI path relative to the one used to access the directory being indexed. If *Filename* begins with a slash, it will be taken to be relative to the **DocumentRoot**.

```
HeaderName /include/HEADER.html
```

Filename must resolve to a document with a major content type of `text/*` (`text/html`, `text/plain`, etc.). This means that *filename* may refer to a CGI script if the script's actual file type (as opposed to its output) is marked as `text/html` such as with a directive like:

```
AddType text/html .cgi
```

[Content negotiation](#) will be performed if [Options MultiViews](#) is in effect. If *filename* resolves to a static text/html document (not a CGI script) and either one of the [options Includes IncludesNOEXEC](#) is enabled, the file will be processed for server-side includes (see the [mod_include](#) documentation).

If the file specified by [HeaderName](#) contains the beginnings of an HTML document (<html>, <head>, etc.) then you will probably want to set [IndexOptions +SuppressHTMLPreamble](#), so that these tags are not repeated.



IndexIgnore

Adds to the list of files to hide when listing a directory
<code>IndexIgnore <i>file</i> [<i>file</i>] ...</code>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

IndexIgnore directive adds to the list of files to hide when listing a directory. *File* is a shell-style wildcard expression or full filename. Multiple **IndexIgnore** directives add to the list, rather than the replacing the list of ignored files. By default, the list contains `.` (the current directory).

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



IndexOptions

Various configuration settings for directory indexing
IndexOptions [+ -]option [[+ -]option] ...
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

IndexOptions directive specifies the behavior of the directory indexing. *Option* can be one of

DescriptionWidth=[*n* | *] (Apache 2.0.23 and later)

The DescriptionWidth keyword allows you to specify the width of the description column in characters.

-DescriptionWidth (or unset) allows [mod_autoindex](#) to calculate the best width.

DescriptionWidth=*n* fixes the column width to *n* bytes wide.

DescriptionWidth=* grows the column to the width necessary to accommodate the longest description string.

See the section on [AddDescription](#) for dangers inherent in truncating descriptions.

FancyIndexing

This turns on fancy indexing of directories.

FoldersFirst (Apache 2.0.23 and later)

If this option is enabled, subdirectory listings will *always* appear first, followed by normal files in the directory. The listing is basically broken into two components, the files and the subdirectories, and each is sorted separately and then displayed subdirectories-first. For instance, if the sort order is descending by name, and FoldersFirst is enabled, subdirectory Zed will be listed before subdirectory Beta, which will be listed before normal files GammaAlpha. **This option only has an effect if**

[FancyIndexing](#) is also enabled.

HTMLTable (*Experimental, Apache 2.0.23 and later*)

This experimental option with FancyIndexing constructs a simple table for the fancy directory listing. Note this will confuse older browsers. It is particularly necessary if file names or description text will alternate between left-to-right and right-to-left reading order, as can happen on WinNT or other utf-8 enabled platforms.

IconsAreLinks

This makes the icons part of the anchor for the filename, for fancy indexing.

IconHeight[=*pixels*]

Presence of this option, when used with IconWidth, will cause the server to include heightwidth attributes in the `img` tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard height of the icons supplied with the Apache software.

IconWidth[=*pixels*]

Presence of this option, when used with IconHeight, will cause the server to include heightwidth attributes in the `img` tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard width of the icons supplied with the Apache software.

IgnoreCase

If this option is enabled, names are sorted in a case-insensitive manner. For instance, if the sort order is ascending by name, and IgnoreCase is enabled, file Zeta will be listed after file alfa (Note: file GAMMA will always be listed before file gamma).

IgnoreClient

This option causes `mod_autoindex` to ignore all query variables from the client, including sort order (implies

[SuppressColumnSorting](#).)

NameWidth=[*n* | *]

The NameWidth keyword allows you to specify the width of the filename column in bytes.

-NameWidth (or unset) allows [mod_autoindex](#) to calculate the best width.

NameWidth=*n* fixes the column width to *n* bytes wide.

NameWidth=* grows the column to the necessary width.

ScanHTMLTitles

This enables the extraction of the title from HTML documents for fancy indexing. If the file does not have a description given by [AddDescription](#) then httpd will read the document for the value of the title element. This is CPU and disk intensive.

ShowForbidden

If specified, Apache will show files normally hidden because the subrequest returned HTTP_UNAUTHORIZED or HTTP_FORBIDDEN

SuppressColumnSorting

If specified, Apache will not make the column headings in a FancyIndexed directory listing into links for sorting. The default behavior is for them to be links; selecting the column heading will sort the directory listing by the values in that column. **Prior to Apache 2.0.23, this also disabled parsing the Query Arguments for the sort string.** That behavior is now controlled by [IndexOptions IgnoreClient](#) in Apache 2.0.23.

SuppressDescription

This will suppress the file description in fancy indexing listings. By default, no file descriptions are defined, and so the use of this option will regain 23 characters of screen space to use for something else. See [AddDescription](#) for information about setting the file description. See also the [DescriptionWidth](#) index option to limit the size of the description column.

SuppressHTMLPreamble

If the directory actually contains a file specified by the [HeaderName](#) directive, the module usually includes the contents of the file after a standard HTML preamble (<html>, <head>, *et cetera*). The SuppressHTMLPreamble option disables this behaviour, causing the module to start the display with the header file contents. The header file must contain appropriate HTML instructions in this case. If there is no header file, the preamble is generated as usual.

SuppressIcon (*Apache 2.0.23 and later*)

This will suppress the icon in fancy indexing listings. Combining both SuppressIconSuppressRules yields proper HTML 3.2 output, which by the final specification prohibits `img` elements from the `pre` block (used to format FancyIndexed listings.)

SuppressLastModified

This will suppress the display of the last modification date, in fancy indexing listings.

SuppressRules (*Apache 2.0.23 and later*)

This will suppress the horizontal rule lines (`hr` elements) in directory listings. Combining both SuppressIcon SuppressRules yields proper HTML 3.2 output, which by the final specification prohibits `img` elements from the `pre` block (used to format FancyIndexed listings.)

SuppressSize

This will suppress the file size in fancy indexing listings.

TrackModified (*Apache 2.0.23 and later*)

This returns the Last-Modified and ETag values for the listed directory in the HTTP header. It is only valid if the operating system and file system return appropriate `stat()` results. Some Unix systems do so, as do OS2's JFS and Win32's NTFS volumes. OS2 and Win32 FAT volumes, for example, do not. Once this feature is enabled, the client or proxy can track

changes to the list of files when they perform a HEAD request. Note some operating systems correctly track new and removed files, but do not track changes for sizes or dates of the files within the directory. **Changes to the size or date stamp of an existing file will not update the Last-Modified header on all Unix platforms.** If this is a concern, leave this option disabled.

VersionSort (*Apache 2.0a3 and later*)

The `VersionSort` keyword causes files containing version numbers to sort in a natural way. Strings are sorted as usual, except that substrings of digits in the name and description are compared according to their numeric value.

```
foo-1.7
foo-1.7.2
foo-1.7.12
foo-1.8.2
foo-1.8.2a
foo-1.12
```

If the number starts with a zero, then it is considered to be a fraction:

```
foo-1.001
foo-1.002
foo-1.030
foo-1.04
```

XHTML (*Apache 2.0.49 and later*)

The `XHTML` keyword forces `mod_autoindex` to emit XHTML 1.0 code instead of HTML 3.2.

Incremental IndexOptions

Apache 1.3.3 introduced some significant changes in the

handling of `IndexOptions` directives. In particular:

- Multiple `IndexOptions` directives for a single directory are now merged together. The result of:

```
<Directory /foo>
  IndexOptions HTMLTable
  IndexOptions SuppressColumnsorting
</Directory>
```

will be the equivalent of

```
IndexOptions HTMLTable
SuppressColumnsorting
```

- The addition of the incremental syntax (*i.e.*, prefixing keywords with `+-`).

Whenever a '+' or '-' prefixed keyword is encountered, it is applied to the current `IndexOptions` settings (which may have been inherited from an upper-level directory). However, whenever an unprefixed keyword is processed, it clears all inherited options and any incremental settings encountered so far. Consider the following example:

```
IndexOptions +ScanHTMLTitles -IconsAreLinks
FancyIndexing
IndexOptions +SuppressSize
```

The net effect is equivalent to `IndexOptions FancyIndexing +SuppressSize`, because the unprefixed `FancyIndexing` discarded the incremental keywords before it, but allowed them to start accumulating again afterward.

To unconditionally set the `IndexOptions` for a particular directory, clearing the inherited settings, specify keywords without any +- prefixes.



Sets the default ordering of the directory index
<code>IndexOrderDefault</code> Ascending Descending Name Date Size Description
<code>IndexOrderDefault</code> Ascending Name
server config, virtual host, directory, .htaccess
Indexes
(B)
<code>mod_autoindex</code>

`IndexOrderDefault` directive is used in combination with the [FancyIndexing](#) index option. By default, fancyindexed directory listings are displayed in ascending order by filename; the `IndexOrderDefault` allows you to change this initial display order.

`IndexOrderDefault` takes two arguments. The first must be either `AscendingDescending`, indicating the direction of the sort. The second argument must be one of the keywords `Name`, `Date`, `Size`, or `Description`, and identifies the primary key. The secondary key is *always* the ascending filename.

You can force a directory listing to only be displayed in a particular order by combining this directive with the [SuppressColumnSorting](#) index option; this will prevent the client from requesting the directory listing in a different order.



IndexStyleSheet

Adds a CSS stylesheet to the directory index
<code>IndexStyleSheet <i>url-path</i></code>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

IndexStyleSheet directive sets the name of the file that will be used as the CSS for the index listing.

```
IndexStyleSheet "/css/style.css"
```



ReadmeName

Name of the file that will be inserted at the end of the index listing
ReadmeName <i>filename</i>
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_autoindex

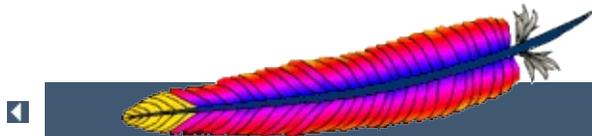
ReadmeName directive sets the name of the file that will be appended to the end of the index listing. *Filename* is the name of the file to include, and is taken to be relative to the location being indexed. If *Filename* begins with a slash, it will be taken to be relative to the [DocumentRoot](#).

```
ReadmeName FOOTER.html
```

Example 2

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

See also [HeaderName](#), where this behavior is described in greater detail.



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_cache

URI()
(E)
cache_module
mod_cache.c

This module should be used with care and can be used to circumvent [AllowDeny](#) directives. You should not enable caching for any content to which you wish to limit access by client host name, address or environment variable.

[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.



<u>mod disk cache</u>	<u>CacheRoot</u>
<u>mod mem cache</u>	<u>CacheSize</u>
	<u>CacheDirLevels</u>
	<u>CacheDirLength</u>
	<u>CacheMinFileSize</u>
	<u>CacheMaxFileSize</u>
	<u>MCacheSize</u>
	<u>MCacheMaxObjectCount</u>
	<u>MCacheMinObjectSize</u>
	<u>MCacheMaxObjectSize</u>
	<u>MCacheRemovalAlgorithm</u>
	<u>MCacheMaxStreamingBuffer</u>



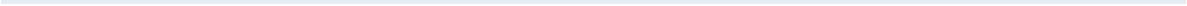
Sample httpd.conf

```
#
# Sample Cache Configuration
#
LoadModule cache_module modules/mod_cache.so

<IfModule mod_cache.c>
  #LoadModule disk_cache_module
  modules/mod_disk_cache.so
  # If you want to use mod_disk_cache instead of
  mod_mem_cache,
  # uncomment the line above and comment out the
  LoadModule line below.
  <IfModule mod_disk_cache.c>
    CacheRoot c:/cacheroot
    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>

  # When acting as a proxy, don't cache the list
  of security updates
  CacheDisable
  http://security.update.server/update-list/
</IfModule>
```



CacheDefaultExpire

The default duration to cache a document when no expiry date is specified.
CacheDefaultExpire <i>seconds</i>
CacheDefaultExpire 3600 (one hour)
server config, virtual host
(E)
mod_cache

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
```



CacheDisable

Disable caching of specified URLs
CacheDisable <i>url-string</i>
server config, virtual host
(E)
mod_cache

CacheDisable directive instructs mod_cache to *not* cache urls at or below *url-string*.

```
CacheDisable /local_files
```



Enable caching of specified URLs using a specified storage manager
<code>CacheEnable <i>cache_type url-string</i></code>
server config, virtual host
(E)
<code>mod_cache</code>

`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 /
```

When acting as a forward proxy server, *url-string* can also be used to specify remote sites and proxy protocols which caching should be enabled for.

```
# Cache proxied url's
```

```
CacheEnable disk /
```

```
# Cache FTP-proxied url's
```

```
CacheEnable disk ftp://
```

```
# Cache content from www.apache.org
```

```
CacheEnable disk http://www.apache.org/
```



Ignore request to not serve cached content to client
CacheIgnoreCacheControl On Off
CacheIgnoreCacheControl Off
server config, virtual host
(E)
mod_cache

Ordinarily, requests containing a Cache-Control: no-cache or Pragma: no-cache header value will not be served from the cache. The `CacheIgnoreCacheControl` directive allows this behavior to be overridden. `CacheIgnoreCacheControl On` tells the server to attempt to serve the resource from the cache even if the request contains no-cache header values. Resources requiring authorization will *never* be cached.

CacheIgnoreCacheControl On

Warning:

This directive will allow serving from the cache even if the client has requested that the document not be served from the cache. This might result in stale content being served.

- [CacheStorePrivate](#)
- [CacheStoreNoStore](#)



CacheIgnoreHeaders

Do not store the given HTTP header(s) in the cache.
CacheIgnoreHeaders <i>header-string</i> [<i>header-string</i>] ...
CacheIgnoreHeaders None
server config, virtual host
(E)
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

Ignore the fact that a response has no Last Modified header.
CacheIgnoreNoLastMod On Off
CacheIgnoreNoLastMod Off
server config, virtual host
(E)
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



CacheLastModifiedFactor

The factor used to compute an expiry date based on the LastModified date.
CacheLastModifiedFactor <i>float</i>
CacheLastModifiedFactor 0.1
server config, virtual host
(E)
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

The maximum time in seconds to cache a document
CacheMaxExpire <i>seconds</i>
CacheMaxExpire 86400 (one day)
server config, virtual host
(E)
mod_cache

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
```



Attempt to cache requests or responses that have been marked as no-store.
CacheStoreNoStore On Off
CacheStoreNoStore Off
server config, virtual host
(E)
mod_cache

Ordinarily, requests or responses with Cache-Control: no-store header values will not be stored in the cache. The `CacheStoreNoCache` directive allows this behavior to be overridden. `CacheStoreNoCache On` tells the server to attempt to cache the resource even if it contains no-store header values. Resources requiring authorization will *never* be cached.

```
CacheStoreNoStore On
```

Warning:

As described in RFC 2616, the no-store directive is intended to "prevent the inadvertent release or retention of sensitive information (for example, on backup tapes)." Enabling this option could store sensitive information in the cache. You are hereby warned.

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



Cache-Control: Private

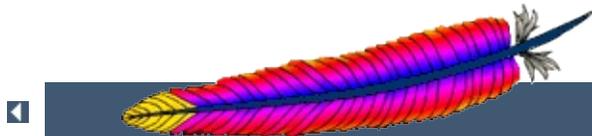
Attempt to cache responses that the server has marked as private
CacheStorePrivate On Off
CacheStorePrivate Off
server config, virtual host
(E)
mod_cache

Ordinarily, responses with Cache-Control: private header values will not be stored in the cache. The `CacheStorePrivate` directive allows this behavior to be overridden. `CacheStorePrivate On` tells the server to attempt to cache the resource even if it contains private header values. Resources requiring authorization will *never* be cached.

CacheStorePrivate On

Warning:
This directive will allow caching even if the upstream server has requested that the resource not be cached. This directive is only ideal for a 'private' cache.

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



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_cern_meta

ApacheCERN httpd
(E)
cern_meta_module
mod_cern_meta.c

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.



Name of the directory to find CERN-style meta information files
MetaDir <i>directory</i>
MetaDir <code>.web</code>
server config, virtual host, directory, <code>.htaccess</code>
Indexes
(E)
<code>mod_cern_meta</code>

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
```



MetaFiles

Activates CERN meta-file processing
MetaFiles on off
MetaFiles off
server config, virtual host, directory, .htaccess
Indexes
(E)
mod_cern_meta

Turns on/off Meta file processing on a per-directory basis.

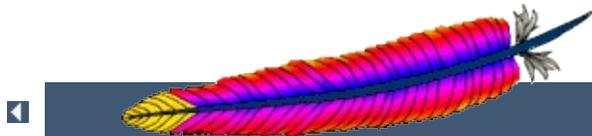


MetaSuffix

File name suffix for the file containing CERN-style meta information
MetaSuffix <i>suffix</i>
MetaSuffix <code>.meta</code>
server config, virtual host, directory, <code>.htaccess</code>
Indexes
(E)
<code>mod_cern_meta</code>

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.

MetaSuffix `.meta`



◀ **Apache HTTP Server 2.2**

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

| | | 2006124 |

Apache mod_cgi

MPM(<u>prefork</u>)CGI
(B)
cgi_module
mod_cgi.c

MIMEapplication/x-httpd-cgicgi-scriptCGICGI

AddType ScriptAlias

CGIDOCUMENT_ROOT DocumentRoot

ApacheCGI CGI

UNIXMPM mod_cgid



ApacheCGI

PATH_INFO

AcceptPathInfo off AcceptPathInfo mod_cgi
(URI /more/path/info)"404 NOT FOUND"
AcceptPathInfo On mod_cgi

REMOTE_HOST

HostnameLookups" on("off")DNS

REMOTE_IDENT

IdentityCheck on

REMOTE_USER

CGI



CGI(stdoutstderr)

CGI

CGICGICGI

```
%% [time] request-line  
%% HTTP-status CGI-script-filename
```

CGI

```
%%error  
error-message
```

(bug)

```
%request  
All HTTP request headers received  
POST or PUT entity (if any)  
%response  
All headers output by the CGI script  
%stdout  
CGI standard output  
%stderr  
CGI standard error
```

stdoutstderr%stdout%stderr



ScriptLog

CGI
ScriptLog <i>file-path</i>
server config, virtual host
(B)
<u>mod_cgi</u> , <u>mod_cgid</u>

ScriptLogCGI

ScriptLog CGI

ServerRoot

```
ScriptLog logs/cgi_log
```

User

CGI



ScriptLogBuffer

PUTPOST
ScriptLogBuffer <i>bytes</i>
ScriptLogBuffer 1024
server config, virtual host
(B)
<u>mod_cgi</u> , <u>mod_cgid</u>

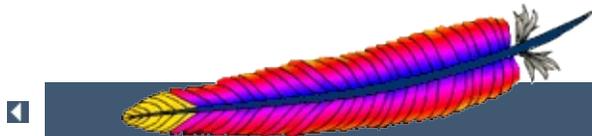
PUTPOST1024



ScriptLogLength

()
ScriptLogLength <i>bytes</i>
ScriptLogLength 10385760
server config, virtual host
(B)
<u>mod_cgi</u> , <u>mod_cgid</u>

ScriptLogLengthCGICGI()CGI



◀ Apache HTTP Server 2.2

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

| | | 2006124 |

Apache mod_cgid

MPM(<u>worker</u>)CGICGI
(B)
cgid_module
mod_cgid.c
UnixMPM

ScriptSock mod_cgidmod_cgi mod_cgiApacheCGI

unixforkCGI mod_cgidforkCGIunix domain

MPM mod_cgi mod_cgi ScriptSockcgi

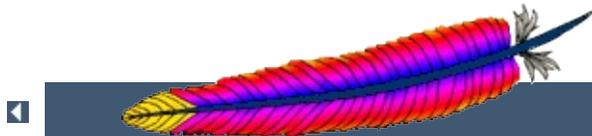


CGISock

CGI
ScriptSock <i>file-path</i>
ScriptSock logs/cgisock
server config, virtual host
(B)
mod_cgid

CGI(PID)Apache(root)CGI

ScriptSock /var/run/cgid.sock



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_charset_lite

(X)
charset_lite_module
mod_charset_lite.c

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](#) [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.



Charset to translate into
CharsetDefault <i>charset</i>
server config, virtual host, directory, .htaccess
FileInfo
(X)
mod_charset_lite

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.

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



CharsetOptions

Configures charset translation behavior
CharsetOptions <i>option</i> [<i>option</i>] ...
CharsetOptions DebugLevel=0 NoImplicitAdd
server config, virtual host, directory, .htaccess
FileInfo
(X)
mod_charset_lite

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.



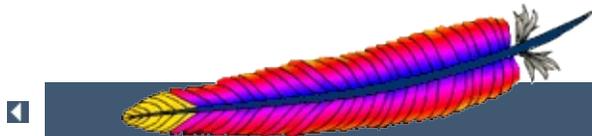
Source charset of files
CharsetSourceEnc <i>charset</i>
server config, virtual host, directory, .htaccess
FileInfo
(X)
mod_charset_lite

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.

```
<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.



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_dav

ApacheDAV
(E)
dav_module
mod_dav.c

This module provides class 1 and class 2 [WebDAV](#) ('Web-based Distributed Authoring and Versioning') functionality for Apache. This extension to the HTTP protocol allows creating, moving, copying, and deleting resources and collections on a remote web server.



To enable `mod_dav`, add the following to a container in your `httpd.conf` file:

```
Dav On
```

This enables the DAV file system provider, which is implemented by the `mod_dav_fs` module. Therefore, that module must be compiled into the server or loaded at runtime using the `LoadModule` directive.

In addition, a location for the DAV lock database must be specified in the global section of your `httpd.conf` file using the `DavLockDB` directive:

```
DavLockDB /usr/local/apache2/var/DavLock
```

The directory containing the lock database file must be writable by the `UserGroup` under which Apache is running.

You may wish to add a `<Limit>` clause inside the `<Location>` directive to limit access to DAV-enabled locations. If you want to set the maximum amount of bytes that a DAV client can send at one request, you have to use the `LimitXMLRequestBody` directive. The "normal" `LimitRequestBody` directive has no effect on DAV requests.

Full Example

```
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](#) is a descendent of Greg Stein's [mod_dav for Apache 1.3](#).
More information about the module is available from that site.



Since DAV access methods allow remote clients to manipulate files on the server, you must take particular care to assure that your server is secure before enabling [mod_dav](#).

Any location on the server where DAV is enabled should be protected by authentication. The use of HTTP Basic Authentication is not recommended. You should use at least HTTP Digest Authentication, which is provided by the [mod_auth_digest](#) module. Nearly all WebDAV clients support this authentication method. An alternative is Basic Authentication over an [SSL](#) enabled connection.

In order for [mod_dav](#) to manage files, it must be able to write to the directories and files under its control using the [UserGroup](#) under which Apache is running. New files created will also be owned by this [UserGroup](#). For this reason, it is important to control access to this account. The DAV repository is considered private to Apache; modifying files outside of Apache (for example using FTP or filesystem-level tools) should not be allowed.

[mod_dav](#) may be subject to various kinds of denial-of-service attacks. The [LimitXMLRequestBody](#) directive can be used to limit the amount of memory consumed in parsing large DAV requests. The [DavDepthInfinity](#) directive can be used to prevent PROPFIND requests on a very large repository from consuming large amounts of memory. Another possible denial-of-service attack involves a client simply filling up all available disk space with many large files. There is no direct way to prevent this in Apache, so you should avoid giving DAV access to untrusted users.



Complex Configurations

One common request is to use `mod_dav` to manipulate dynamic files (PHP scripts, CGI scripts, etc). This is difficult because a GET request will always run the script, rather than downloading its contents. One way to avoid this is to map two different URLs to the content, one of which will run the script, and one of which will allow it to be downloaded and manipulated with DAV.

```
Alias /phparea /home/gstein/php_files
Alias /php-source /home/gstein/php_files
<Location /php-source>
    DAV On
    ForceType text/plain
</Location>
```

With this setup, `http://example.com/phparea` can be used to access the output of the PHP scripts, and `http://example.com/php-source` can be used with a DAV client to manipulate them.



Enable WebDAV HTTP methods
Dav On Off <i>provider-name</i>
Dav Off
directory
(E)
mod_dav

Use the **Dav** directive to enable the WebDAV HTTP methods for the given container:

```
<Location /foo>  
  Dav On  
</Location>
```

The value `On` is actually an alias for the default provider `filesystem` which is served by the `mod_dav_fs` module. Note, that once you have DAV enabled for some location, it *cannot* be disabled for sublocations. For a complete configuration example have a look at the [section above](#).

Do not enable WebDAV until you have secured your server. Otherwise everyone will be able to distribute files on your system.



DavDepthInfinity

Allow PROPFIND, Depth: Infinity requests
DavDepthInfinity on off
DavDepthInfinity off
server config, virtual host, directory
(E)
mod_dav

Use the `DavDepthInfinity` directive to allow the processing of PROPFIND requests containing the header 'Depth: Infinity'. Because this type of request could constitute a denial-of-service attack, by default it is not allowed.

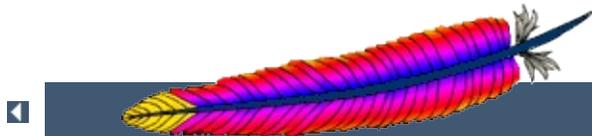


Minimum amount of time the server holds a lock on a DAV resource
<code>DavMinTimeout</code> <i>seconds</i>
<code>DavMinTimeout</code> 0
server config, virtual host, directory
(E)
mod_dav

When a client requests a DAV resource lock, it can also specify a time when the lock will be automatically removed by the server. This value is only a request, and the server can ignore it or inform the client of an arbitrary value.

Use the `DavMinTimeout` directive to specify, in seconds, the minimum lock timeout to return to a client. Microsoft Web Folders defaults to a timeout of 120 seconds; the `DavMinTimeout` can override this to a higher value (like 600 seconds) to reduce the chance of the client losing the lock due to network latency.

```
<Location /MSWord>
  DavMinTimeout 600
</Location>
```



||||

◀ Apache HTTP Server 2.2

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

|| < > | ??? |

Apache mod_dav_fs

mod_dav
(E)
dav_fs_module
mod_dav_fs.c

This module *requires* the service of [mod_dav](#). It acts as a support module for [mod_dav](#) and provides access to resources located in the server's file system. The formal name of this provider is `filesystem`. [mod_dav](#) backend providers will be invoked by using the [Dav](#) directive:

```
Dav filesystem
```

Since `filesystem` is the default provider for [mod_dav](#), you may simply use the value `On` instead.

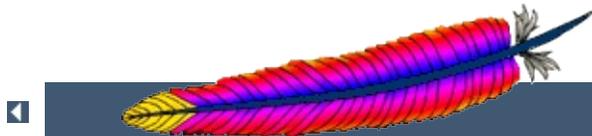


Location of the DAV lock database
DavLockDB <i>file-path</i>
server config, virtual host
(E)
mod_dav_fs

Use the `DavLockDB` directive to specify the full path to the lock database, excluding an extension. If the path is not absolute, it will be taken relative to `ServerRoot`. The implementation of `mod_dav_fs` uses a SDBM database to track user locks.

```
DavLockDB var/DavLock
```

The directory containing the lock database file must be writable by the `UserGroup` under which Apache is running. For security reasons, you should create a directory for this purpose rather than changing the permissions on an existing directory. In the above example, Apache will create files in the `var/` directory under the `ServerRoot` with the base filename `DavLock` and extension name chosen by the server.



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_dav_lock

mod_dav
(E)
dav_lock_module
mod_dav_lock.c
Apache 2.1

This module implements a generic locking API which can be used by any backend provider of [mod_dav](#). It *requires* at least the service of [mod_dav](#). But without a backend provider which makes use of it, it's useless and should not be loaded into the server. A sample backend module which actually utilizes [mod_dav_lock](#), is [mod_dav_svn](#), the subversion provider module.

Note that [mod_dav_fs](#) does *not* need this generic locking module, because it uses it's own more specialized version.

In order to make [mod_dav_lock](#) functional, you just have to specify the location of the lock database using the [DavGenericLockDB](#) directive described below.

Developer's Note

In order to retrieve the pointer to the locking provider function, you have to use the `ap_lookup_provider` API with the arguments `dav-lock`, `generic0`.

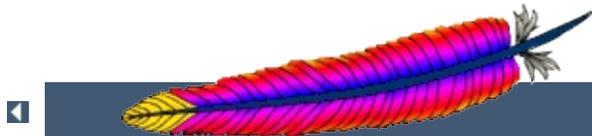


Location of the DAV lock database
<code>DavGenericLockDB</code> <i>file-path</i>
server config, virtual host, directory
(E)
<code>mod_dav_lock</code>

Use the `DavGenericLockDB` directive to specify the full path to the lock database, excluding an extension. If the path is not absolute, it will be taken relative to `ServerRoot`. The implementation of `mod_dav_lock` uses a SDBM database to track user locks.

```
DavGenericLockDB var/DavLock
```

The directory containing the lock database file must be writable by the `UserGroup` under which Apache is running. For security reasons, you should create a directory for this purpose rather than changing the permissions on an existing directory. In the above example, Apache will create files in the `var /` directory under the `ServerRoot` with the base filename `DavLock` and extension name chosen by the server.



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_dbd

SQL
(E)
dbd_module
mod_dbd.c
Version 2.1

[mod_dbd](#) manages SQL database connections using [apr_dbd](#). It provides database connections on request to modules requiring SQL database functions, and takes care of managing databases with optimal efficiency and scalability for both threaded and non-threaded MPMs.



This module manages database connections, in a manner optimised for the platform. On non-threaded platforms, it provides a persistent connection in the manner of classic LAMP (Linux, Apache, Mysql, Perl/PHP/Python). On threaded platform, it provides an altogether more scalable and efficient *connection pool*, as described in [this article at ApacheTutor](#). `mod_dbd` supersedes the modules presented in that article.



`mod_dbd` exports five functions for other modules to use. The API is as follows:

```
typedef struct {
    apr_dbd_t *handle;
    apr_dbd_driver_t *driver;
    apr_hash_t *prepared;
} ap_dbd_t;

/* Export functions to access the database */

/* acquire a connection that MUST be explicitly closed
 * Returns NULL on error
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_open(apr_pool_t*, server_rec*);

/* release a connection acquired with ap_dbd_open */
AP_DECLARE(void) ap_dbd_close(server_rec*, ap_dbd_t*);

/* acquire a connection that will have the lifetime of the request
 * and MUST NOT be explicitly closed. Return NULL on error
 * This is the preferred function for most applications
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_acquire(request_rec*);

/* acquire a connection that will have the lifetime of the request
 * and MUST NOT be explicitly closed. Return NULL on error
 */
AP_DECLARE(ap_dbd_t*) ap_dbd_cacquire(request_rec*);

/* Prepare a statement for use by a client module */
AP_DECLARE(void) ap_dbd_prepare(server_rec*, const char*);

/* Also export them as optional functions for modules
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_open, (apr_pool_t*, server_rec*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_close, (server_rec*, ap_dbd_t*));
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_acquire, (request_rec*));
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_cacquire, (request_rec*));
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepare, (server_rec*, const char*));
```

```
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_acquire, (re  
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_cacquire, (c  
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepare, (server_
```



SQL Prepared Statements

`mod_dbd` supports SQL prepared statements on behalf of modules that may wish to use them. Each prepared statement must be assigned a name (label), and they are stored in a hash: the `prepared` field of an `ap_dbd_t`. Hash entries are of type `apr_dbd_prepared_t` and can be used in any of the `apr_dbd_prepared` statement SQL query or select commands.

It is up to dbd user modules to use the prepared statements and document what statements can be specified in `httpd.conf`, or to provide their own directives and use `ap_dbd_prepare`.



DBDExptime

Keepalive time for idle connections
DBDExptime <i>time-in-seconds</i>
server config, virtual host
(E)
mod_dbd

Set the time to keep idle connections alive where the number of connections specified in DBDKeep has been exceeded (threaded platforms only).



Maximum sustained number of connections
DBDKeep <i>number</i>
server config, virtual host
(E)
mod_dbd

Set the maximum number of connections per process to be sustained, other than for handling peak demand (threaded platforms only).



Maximum number of connections
DBDMax <i>number</i>
server config, virtual host
(E)
mod_dbd

Set the hard maximum number of connections per process (threaded platforms only).



Minimum number of connections
DBDMin <i>number</i>
server config, virtual host
(E)
mod_dbd

Set the minimum number of connections per process (threaded platforms only).



Parameters for database connection
DBDParams <i>param1=value1[,param2=value2]</i>
server config, virtual host
(E)
mod_dbd

As required by the underlying driver. Typically this will be used to pass whatever cannot be defaulted amongst username, password, database name, hostname and port number for connection.



DBD Persistent Connections

Whether to use persistent connections
<code>DBDPersist 0 1</code>
server config, virtual host
(E)
<code>mod_dbd</code>

If set to 0, persistent and pooled connections are disabled. A new database connection is opened when requested by a client, and closed immediately on release. This option is for debugging and low-usage servers.

The default is to enable a pool of persistent connections (or a single LAMP-style persistent connection in the case of a non-threaded server), and should almost always be used in operation.



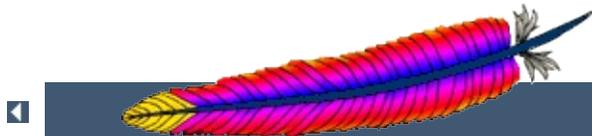
Define an SQL prepared statement
<code>DBDPrepareSQL "SQL statement" label</code>
server config, virtual host
(E)
mod_dbd

For modules such as authentication that use repeatedly use a single SQL statement, optimum performance is achieved by preparing the statement at startup rather than every time it is used. This directive prepares an SQL statement and assigns it a label.



Specify an SQL driver
<code>DBDriver <i>name</i></code>
server config, virtual host
(E)
<code>mod_dbd</code>

Selects an `apr_dbd` driver by name. The driver must be installed on your system (on most systems, it will be a shared object or dll). For example, `DBDriver mysql` will select the MySQL driver in `apr_dbd_mysql.so`.



Apache HTTP Server 2.2

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

| | | 2006125 |

Apache mod_deflate

(E)
deflate_module
mod_deflate.c

[mod_deflate](#)DEFLATE



```
AddOutputFilterByType DEFLATE text/html text/plain
text/xml
```

Compress everything except images

```
<Location />
#
SetOutputFilter DEFLATE

# Netscape 4.x ...
BrowserMatch ^Mozilla/4 gzip-only-text/html

# Netscape 4.06-4.08
BrowserMatch ^Mozilla/4\.0[678] no-gzip

# MSIE Netscape
BrowserMatch \bMSIE !no-gzip !gzip-only-
text/html
#
SetEnvIfNoCase Request_URI \
  \.(?:gif|jpe?g|png)$ no-gzip dont-vary

#
Header append Vary User-Agent env=!dont-vary
</Location>
```



DEFLATE

```
SetOutputFilter DEFLATE
```

```
gzip-only-text/html" 1"html() "1"
```

MIME [AddOutputFilterByType](#)html

```
<Directory "/your-server-root/manual">  
  AddOutputFilterByType DEFLATE text/html  
</Directory>
```

```
BrowserMatchno-gzip no-gzipgzip-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-AgentNavigator 4.x text/html4.06, 4.07, 4.08  
Navigator
```

```
BrowserMatchIE"Mozilla/4" User-Agent"MSIE"(" \b"  
")
```

```
DEFLATEPHPSSI
```

```
SetEnvforce-gzip"accept-encoding"
```

`mod_deflate` `gzip` [SetOutputFilterAddOutputFilter](#)
INFLATE

```
<Location /dav-area>  
  ProxyPass http://example.com/  
  SetOutputFilter INFLATE  
</Location>
```

example.com

`mod_deflate` `gzip` [SetInputFilterAddInputFilter](#) DEFLATE

```
<Location /dav-area>  
  SetInputFilter DEFLATE  
</Location>
```

" Content-Encoding: gzip"

[WebDAV](#)

Content-Length

Content-Length



```
mod_deflate" Vary: Accept-Encoding"HTTP" Accept-  
Encoding"
```

```
( User-Agent) Vary DEFLATEUser-Agent
```

```
Header append Vary User-Agent
```

```
(HTTP) Vary" *"
```

```
Header set Vary *
```



zlib()
DeflateBufferSize <i>value</i>
DeflateBufferSize 8096
server config, virtual host
(E)
mod_deflate

DeflateBufferSizezlib



DeflateCompressionLevel

DeflateCompressionLevel <i>value</i>
Zlib
server config, virtual host
(E)
mod_deflate
Apache 2.0.45

DeflateCompressionLevelCPU

1) 9)



DeflateFilterNote [<i>type</i>] <i>notename</i>
server config, virtual host
(E)
mod_deflate
<i>type</i> 2.0.45

DeflateFilterNote *notename*

DeflateFilterNote ratio

LogFormat '"%r" %b (%{ratio}n) "%{User-agent}i"'

deflate

CustomLog logs/deflate_log deflate

typenotename *type*

Input

Output

Ratio

(/*100) *type*

Accurate Logging

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](#)



zlib
DeflateMemLevel <i>value</i>
DeflateMemLevel 9
server config, virtual host
(E)
mod_deflate

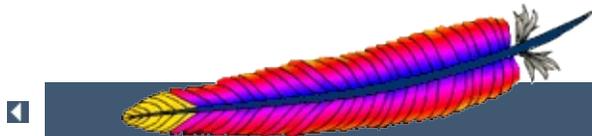
DeflateMemLevelzlib(19)



Zlib(compression window)
DeflateWindowSize <i>value</i>
DeflateWindowSize 15
server config, virtual host
(E)
mod_deflate

DeflateWindowSizezlib(compression window)(115)

| | | |



Apache HTTP Server 2.2

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

| | | 2006125 |

Apache mod_dir

""
(B)
dir_module
mod_dir.c

- `index.html` `mod_dir` `DirectoryIndex`
- `mod_autoindex`

"/" `http://servername/foo/dirname` `dirname`
`mod_dir` `http://servername/foo/dirname/`



DirectoryIndex

DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...
DirectoryIndex index.html
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_dir

DirectoryIndex"/" Local-url(%)URL()URL
Indexes

```
DirectoryIndex index.html
```

http://myserver/docs/
http://myserver/docs/index.html()

URL

```
DirectoryIndex index.html index.txt /cgi-  
bin/index.pl
```

index.htmlindex.txtCGI/cgi-bin/index.pl



DirectorySlash

(/)
DirectorySlash On Off
DirectorySlash On
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_dir
Apache 2.0.51

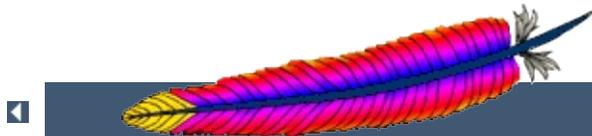
DirectorySlash mod_dir URL "/"

"/" mod_dir URL "/"

- URL
- mod_autoindex
- DirectoryIndex "/"
- htmlURL

```
#  
<Location /some/path>  
  DirectorySlash Off  
  SetHandler some-handler  
</Location>
```

```
mod_autoindex(Options +Indexes)DirectoryIndex(  
index.html)URL "/"URL      index.html  "/"
```



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_disk_cache

(E)
disk_cache_module
mod_disk_cache.c

[mod_disk_cache](#) implements a disk based storage manager. It is primarily of use in conjunction [mod_cache](#).

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.

[htcacheclean](#) can be used to maintain the cache size at a maximum level.

[mod_disk_cache](#) requires the services of [mod_cache](#).



CacheDirLength

The number of characters in subdirectory names
CacheDirLength <i>length</i>
CacheDirLength 2
server config, virtual host
(E)
mod_disk_cache

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



CACHEDIRLEVELS

The number of levels of subdirectories in the cache.
CacheDirLevels <i>levels</i>
CacheDirLevels 3
server config, virtual host
(E)
mod_disk_cache

`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



CACHEMAXFILESIZE

The maximum size (in bytes) of a document to be placed in the cache
<code>CacheMaxFileSize</code> <i>bytes</i>
<code>CacheMaxFileSize</code> 1000000
server config, virtual host
(E)
<code>mod_disk_cache</code>

`CacheMaxFileSize` directive sets the maximum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMaxFileSize 64000
```



CACHEMINFILESIZE

The minimum size (in bytes) of a document to be placed in the cache
<code>CacheMinFileSize</code> <i>bytes</i>
<code>CacheMinFileSize 1</code>
server config, virtual host
(E)
<code>mod_disk_cache</code>

`CacheMinFileSize` directive sets the minimum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMinFileSize 64
```

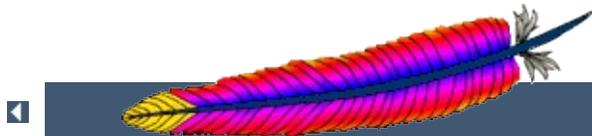


CACHEROOT

The directory root under which cache files are stored
CacheRoot <i>directory</i>
server config, virtual host
(E)
mod_disk_cache

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 `CacheDirLevelsCacheDirLength` directives define the structure of the directories under the specified root directory.

```
CacheRoot c:/cacheroot
```



Apache HTTP Server 2.2

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

| | | 2006125 |

Apache mod_dumpio

I/O
(E)
dumpio_module
mod_dumpio.c

mod_dumpioApache(error.log)

SSL()SSL()





DumpIOInput

DumpIOInput On Off
DumpIOInput Off
server config
(E)
mod_dumpio
Apache 2.1.3

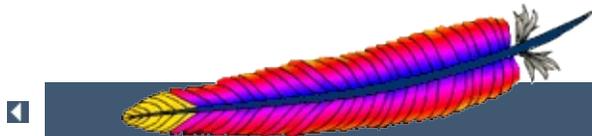
DumpIOInput On



DumpIOOutput

DumpIOOutput On Off
DumpIOOutput Off
server config
(E)
mod_dumpio
Apache 2.1.3

DumpIOOutput On



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_echo

(X)
echo_module
mod_echo.c
Apache 2.0

This module provides an example protocol module to illustrate the concept. It provides a simple echo server. Telnet to it and type stuff, and it will echo it.

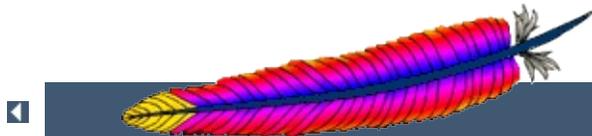


ProtocolEcho

Turn the echo server on or off
ProtocolEcho On Off
server config, virtual host
(X)
mod_echo
ProtocolEcho is only available in 2.0

ProtocolEcho directive enables or disables the echo server.

```
ProtocolEcho On
```



Apache HTTP Server 2.2

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

| | | 2006125 |

Apache mod_env

ApacheCGISSI
(B)
env_module
mod_env.c

CGISSI [httpdshell](#) (set)(unset)



PassEnv

shell
PassEnv <i>env-variable</i> [<i>env-variable</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_env

[httpdshellCGISSI](#)

PassEnv LD_LIBRARY_PATH



CGI

<code>SetEnv env-variable value</code>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_env

CGISSI

```
SetEnv SPECIAL_PATH /foo/bin
```

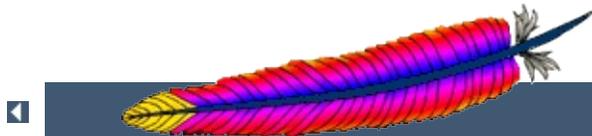


CGISSL

<code>UnsetEnv env-variable [env-variable] ...</code>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_env

CGISSI

`UnsetEnv LD_LIBRARY_PATH`



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_example

ApacheAPI
(X)
example_module
mod_example.c

Some files in the `modules/experimental` 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.



Configuring the Example Module

To include the example module in your server, follow the steps below:

1. Run `configure` with `--enable-example` option.
2. Make the server (run "make").

To add another module of your own:

- A. `cp modules/experimental/mod_example.c modules/new_module/mod_myexample.c`
- B. Modify the file.
- C. Create `modules/new_module/config.m4`.
 1. Add `APACHE_MODPATH_INIT(new_module)`.
 2. Copy `APACHE_MODULE` line with "example" from `modules/experimental/config.m4`.
 3. Replace the first argument "example" with *myexample*.
 4. Replace the second argument with brief description of your module. It will be used in `configure --help`.
 5. If your module needs additional C compiler flags, linker flags or libraries, add them to `CFLAGS`, `LD_FLAGS` and `LIBS` accordingly. See other `config.m4` files in `modules` directory for examples.
 6. Add `APACHE_MODPATH_FINISH`.
- D. Create `module/new_module/Makefile.in`. If your module doesn't need special build instructions, all you need to have in that file is `include $(top_srcdir)/build/special.mk`.
- E. Run `./buildconf` from the top-level directory.
- F. Build the server with `--enable-myexample`



To activate the example module, include a block similar to the following in your `httpd.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
```

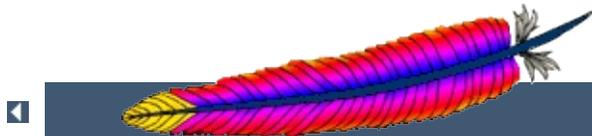
After reloading/restarting your server, you should be able to browse to this location and see the brief display mentioned earlier.



Example

Demonstration directive to illustrate the Apache module API
Example
server config, virtual host, directory, .htaccess
(X)
mod_example

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".



Apache HTTP Server 2.2

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

| | | 2006125 |

Apache mod_expires

HTTP Expires Cache-Control
(E)
expires_module
mod_expires.c

ExpiresCache-Controlmax-age(expiration date)

HTTP()

Cache-Controlmax-age([RFC 2616 section 14.9](#)) [Header](#)



ExpiresDefaultExpiresByType

```
ExpiresDefault "<base> [plus] {<num> <type>}*"
ExpiresByType type/encoding "<base> [plus] {<num>
<type>}*"
```

<base>

- access
- now(' access')
- modification

plus<num>[atoi()]<type>

- years
- months
- weeks
- days
- hours
- minutes
- seconds

3

```
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:"

""



ExpiresCache-Control

" Expires:"" Cache-Control:"
ExpiresActive On Off
server config, virtual host, directory, .htaccess
Indexes
(E)
mod_expires

ExpiresCache-Control Off ExpiresCache-Control(
.htaccess) On ExpiresByTypeExpiresDefault
ExpiresCache-Control

ExpiresCache-Control



ExpiresByType

MIMEExpires
ExpiresByType <i>MIME-type</i> <code><code>seconds</code>
server config, virtual host, directory, .htaccess
Indexes
(E)
mod_expires

MIME(text/html)ExpiresCache-Controlmax-age seconds
Cache-Control: max-age

```
<code>" M"" A" <code>seconds  
" M"URL()" A"
```

```
#  
ExpiresActive On  
# GIF1  
ExpiresByType image/gif A2592000  
# HTML  
ExpiresByType text/html M604800
```

" ExpiresActive On" MIMEExpiresDefault

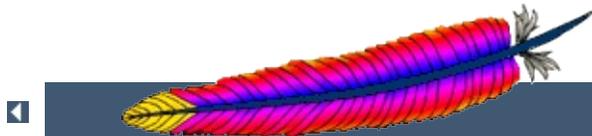
[alternate syntax](#)



ExpiresDefault

ExpiresDefault <i><code>seconds</i>
server config, virtual host, directory, .htaccess
Indexes
(E)
mod_expires

[ExpiresByTypeMIME](#) [ExpiresByTypealternate syntax](#)



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_ext_filter

(E)
ext_filter_module
mod_ext_filter.c

[mod_ext_filter](#) presents a simple and familiar programming model for . 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.



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);
```



Define an external filter
<code>ExtFilterDefine <i>filtername parameters</i></code>
server config
(E)
<code>mod_ext_filter</code>

`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 (`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`

Use `mode=output` (the default) for filters which process the response. Use `mode=input` for filters which process the request. `mode=input` is available in Apache 2.1 and later.

`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.



Configuration Examples

Configure <code>mod_ext_filter</code> options
<code>ExtFilterOptions option [option] ...</code>
<code>ExtFilterOptions DebugLevel=0 NoLogStderr</code>
directory
(E)
<code>mod_ext_filter</code>

`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

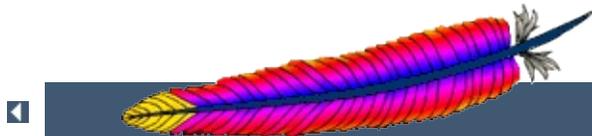
The `LogStderr` keyword specifies that messages written to standard error by the external filter program will be saved in the Apache error log. `NoLogStderr` disables this feature.

```
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.

||||



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_file_cache

Apache
(X)
file_cache_module
mod_file_cache.c

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.

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 `MMapFileCacheFile` 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

`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` `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

`CacheFile` directive of `mod_file_cache` opens an active *handle*

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` `mv` do this.

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
```



CacheFile

Cache a list of file handles at startup time
CacheFile <i>file-path</i> [<i>file-path</i>] ...
server config
(X)
mod_file_cache

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](#).

```
CacheFile /usr/local/apache/htdocs/index.html
```



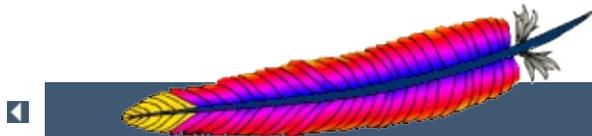
MMapFile

Map a list of files into memory at startup time
<code>MMapFile file-path [file-path] ...</code>
server config
(X)
<code>mod_file_cache</code>

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 HUPUSR1 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](#)[mod_rewrite](#).

```
MMapFile /usr/local/apache/htdocs/index.html
```



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_filter

(B)
filter_module
mod_filter.c
Version 2.1

This module enables smart, context-sensitive configuration of output content filters. For example, apache can be configured to process different content-types through different filters, even when the content-type is not known in advance (e.g. in a proxy).

mod_filter works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to mod_filter; no change to existing filter modules is required (although it may be possible to simplify them).



Content Filtering

In the traditional filtering model, filters are inserted unconditionally using [AddOutputFilter](#) and family. Each filter then needs to determine whether to run, and there is little flexibility available for server admins to allow the chain to be configured dynamically.

[mod_filter](#) by contrast gives server administrators a great deal of flexibility in configuring the filter chain. In fact, filters can be inserted based on any Request Header, Response Header or Environment Variable. This generalises the limited flexibility offered by [AddOutputFilterByType](#), and fixes it to work correctly with dynamic content, regardless of the content generator. The ability to dispatch based on Environment Variables offers the full flexibility of configuration with [mod_rewrite](#) to anyone who needs it.



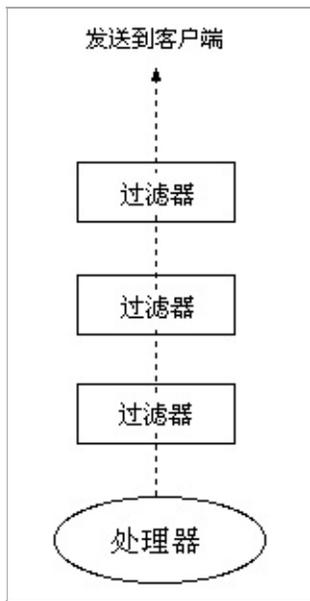


Figure 1: The traditional filter model

In the traditional model, output filters are a simple chain from the content generator (handler) to the client. This works well provided the filter chain can be correctly configured, but presents problems when the filters need to be configured dynamically based on the outcome of the handler.

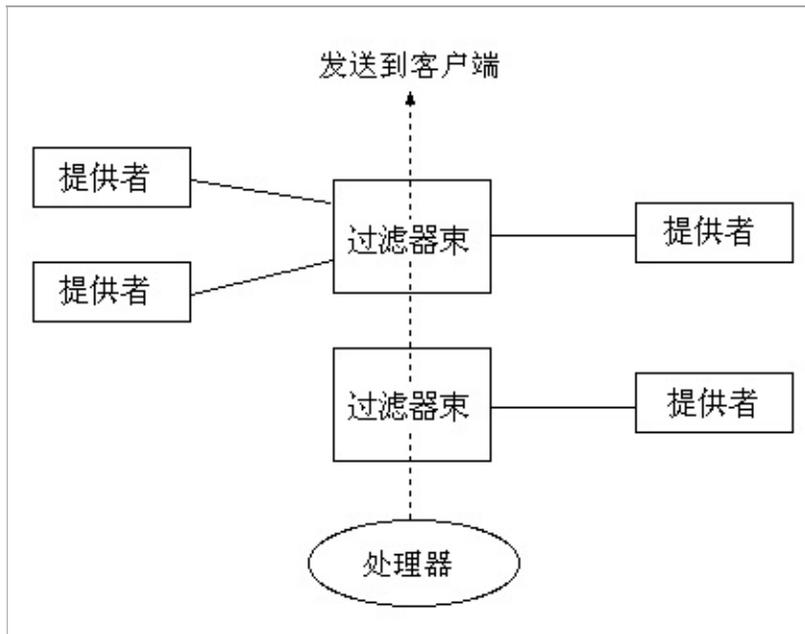


Figure 2: The *mod filter* model

mod filter works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to mod filter; no change to existing filter modules is required (although it may be possible to simplify them). There can be multiple providers for one filter, but no more than one provider will run for any single request.

A filter chain comprises any number of instances of the filter harness, each of which may have any number of providers. A special case is that of a single provider with unconditional dispatch: this is equivalent to inserting the provider filter directly into the chain.



There are three stages to configuring a filter chain with `mod_filter`. For details of the directives, see below.

Declare Filters

The `FilterDeclare` directive declares a filter, assigning it a name and filter type. Required only if the filter is not the default type `AP_FTYPE_RESOURCE`.

Register Providers

The `FilterProvider` directive registers a provider with a filter. The filter may have been declared with `FilterDeclare`; if not, `FilterProvider` will implicitly declare it with the default type `AP_FTYPE_RESOURCE`. The provider must have been registered with `ap_register_output_filter` by some module. The remaining arguments to `FilterProvider` are a dispatch criterion and a match string. The former may be an HTTP request or response header, an environment variable, or the Handler used by this request. The latter is matched to it for each request, to determine whether this provider will be used to implement the filter for this request.

Configure the Chain

The above directives build components of a smart filter chain, but do not configure it to run. The `FilterChain` directive builds a filter chain from smart filters declared, offering the flexibility to insert filters at the beginning or end of the chain, remove a filter, or clear the chain.



Server side Includes (SSI)

A simple case of using `mod_filter` in place of `AddOutputFilterByType`

```
FilterDeclare SSI
FilterProvider SSI INCLUDES resp=Content-Type
$text/html
FilterChain SSI
```

Server side Includes (SSI)

The same as the above but dispatching on handler (classic SSI behaviour; .shtml files get processed).

```
FilterProvider SSI INCLUDES Handler server-
parsed
FilterChain SSI
```

Emulating `mod_gzip` with `mod_deflate`

Insert INFLATE filter only if "gzip" is NOT in the Accept-Encoding header. This filter runs with ftype CONTENT_SET.

```
FilterDeclare gzip CONTENT_SET
FilterProvider gzip inflate req=Accept-
Encoding !$gzip
FilterChain gzip
```

Image Downsampling

Suppose we want to downsample all web images, and have filters for GIF, JPEG and PNG.

```
FilterProvider unpack jpeg_unpack Content-Type
$image/jpeg
FilterProvider unpack gif_unpack Content-Type
```

```
$image/gif
FilterProvider unpack png_unpack Content-Type
$image/png

FilterProvider downsample downsample_filter
Content-Type $image
FilterProtocol downsample "change=yes"

FilterProvider repack jpeg_pack Content-Type
$image/jpeg
FilterProvider repack gif_pack Content-Type
$image/gif
FilterProvider repack png_pack Content-Type
$image/png
<Location /image-filter>
  FilterChain unpack downsample repack
</Location>
```



Filtering

Historically, each filter is responsible for ensuring that whatever changes it makes are correctly represented in the HTTP response headers, and that it does not run when it would make an illegal change. This imposes a burden on filter authors to re-implement some common functionality in every filter:

- Many filters will change the content, invalidating existing content tags, checksums, hashes, and lengths.
- Filters that require an entire, unbroken response in input need to ensure they don't get byteranges from a backend.
- Filters that transform output in a filter need to ensure they don't violate a `Cache-Control: no-transform` header from the backend.
- Filters may make responses uncacheable.

`mod_filter` aims to offer generic handling of these details of filter implementation, reducing the complexity required of content filter modules. This is work-in-progress; the `FilterProtocol` implements some of this functionality for back-compatibility with Apache 2.0 modules. For httpd 2.1 and later, the `ap_register_output_filter_protocol` and `ap_filter_protocol` API enables filter modules to declare their own behaviour.

At the same time, `mod_filter` should not interfere with a filter that wants to handle all aspects of the protocol. By default (i.e. in the absence of any `FilterProtocol` directives), `mod_filter` will leave the headers untouched.

At the time of writing, this feature is largely untested, as modules in common use are designed to work with 2.0. Modules using it should test it carefully.



FilterChain

Configure the filter chain
<code>FilterChain [+=-@!]filter-name ...</code>
server config, virtual host, directory, .htaccess
Options
(B)
mod_filter

This configures an actual filter chain, from declared filters.

FilterChain takes any number of arguments, each optionally preceded with a single-character control that determines what to do:

+*filter-name*

Add *filter-name* to the end of the filter chain

@*filter-name*

Insert *filter-name* at the start of the filter chain

-*filter-name*

Remove *filter-name* from the filter chain

=*filter-name*

Empty the filter chain and insert *filter-name*

!

Empty the filter chain

filter-name

Equivalent to +*filter-name*



FilterDeclare

Declare a smart filter
<code>FilterDeclare <i>filter-name</i> [<i>type</i>]</code>
server config, virtual host, directory, .htaccess
Options
(B)
mod_filter

This directive declares an output filter together with a header or environment variable that will determine runtime configuration. The first argument is a *filter-name* for use in [FilterProvider](#), [FilterChainFilterProtocol](#) directives.

The final (optional) argument is the type of filter, and takes values of `ap_filter_type` - namely RESOURCE (the default), CONTENT_SET, PROTOCOL, TRANSCODE, CONNECTIONNETWORK.



Deal with correct HTTP protocol handling
<code>FilterProtocol <i>filter-name</i> [<i>provider-name</i>] <i>proto-flags</i></code>
server config, virtual host, directory, .htaccess
Options
(B)
mod_filter

This directs [mod_filter](#) to deal with ensuring the filter doesn't run when it shouldn't, and that the HTTP response headers are correctly set taking into account the effects of the filter.

There are two forms of this directive. With three arguments, it applies specifically to a *filter-name* and a *provider-name* for that filter. With two arguments it applies to a *filter-name* whenever the filter runs *any* provider.

proto-flags is one or more of

change=yes

The filter changes the content, including possibly the content length

change=1:1

The filter changes the content, but will not change the content length

byteranges=no

The filter cannot work on byteranges and requires complete input

proxy=no

The filter should not run in a proxy context

proxy=transform

The filter transforms the response in a manner incompatible with the HTTP Cache-Control: no-transform header.

cache=no

The filter renders the output uncacheable (eg by introducing randomised content changes)



Filter Provider

Register a content filter
<code>FilterProvider <i>filter-name</i> <i>provider-name</i> [req resp env]=<i>dispatch</i> <i>match</i></code>
server config, virtual host, directory, .htaccess
Options
(B)
<code>mod_filter</code>

This directive registers a *provider* for the smart filter. The provider will be called if and only if the *match* declared here matches the value of the header or environment variable declared as *dispatch*.

provider-name must have been registered by loading a module that registers the name with `ap_register_output_filter`.

dispatch argument is a string with optional req=, resp=env= prefix causing it to dispatch on (respectively) the request header, response header, or environment variable named. In the absence of a prefix, it defaults to a response header. A special case is the word `handler`, which causes `mod_filter` to dispatch on the content handler.

match argument specifies a match that will be applied to the filter's *dispatch* criterion. The *match* may be a string match (exact match or substring), a `regex`, an integer (greater, less than or equals), or unconditional. The first characters of the *match* argument determines this:

First, if the first character is an exclamation mark (!), this reverses the rule, so the provider will be used if and only if the match *fails*.

Second, it interprets the first character excluding any leading ! as follows:

--	--

Character	Description
<i>(none)</i>	exact match
\$	substring match
/	regex match (delimited by a second /)
=	integer equality
<	integer less-than
<=	integer less-than or equal
>	integer greater-than
>=	integer greater-than or equal
*	Unconditional match



Get debug/diagnostic information from mod_filter
FilterTrace <i>filter-name level</i>
server config, virtual host, directory
(B)
mod_filter

This directive generates debug information from [mod_filter](#). It is designed to help test and debug providers (filter modules), although it may also help with [mod_filter](#) itself.

The debug output depends on the *level* set:

0 (default)

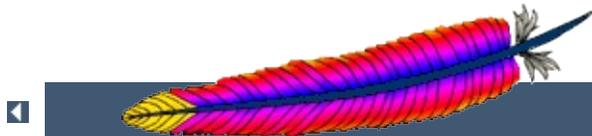
No debug information is generated.

1

[mod_filter](#) will record buckets and brigades passing through the filter to the error log, before the provider has processed them. This is similar to the information generated by [mod_diagnostics](#).

2 (not yet implemented)

Will dump the full data passing through to a tempfile before the provider. **For single-user debug only**; this will not support concurrent hits.



Apache HTTP Server 2.2

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

| | | 2006125 |

Apache mod_headers

HTTP
(E)
headers_module
mod_headers.c
RequestHeader Apache 2.0

HTTP



mod_headers

```
RequestHeader append MirrorID "mirror 12"  
RequestHeader unset MirrorID
```

```
MirrorID      MirrorID"mirror 12"
```



`mod_headers` "[when Request Headers are set immediately before running the content generator and Response Headers just as the response is sent down the wire.]"

`"/ early"`

`"URL" <Directory><Location>`



1. "TS"

```
Header echo ^TS
```

2. MyHeader

```
Header add MyHeader "%D %t"
```

```
MyHeader: D=3775428 t=991424704447256
```

3. Joe>Hello)

```
Header add MyHeader "Hello Joe. It took %D  
microseconds \  
for Apache to serve this request."
```

```
MyHeader: Hello Joe. It took D=3775428  
microseconds for Apache to serve this request.
```

4. "MyRequestHeader" MyHeader" [mod_setenvif](#)

```
SetEnvIf MyRequestHeader value  
HAVE_MyRequestHeader  
Header add MyHeader "%D %t mytext"  
env=HAVE_MyRequestHeader
```

```
" MyRequestHeader: value"
```

MyHeader: D=3775428 t=991424704447256 mytext



HTTP
Header [<i>condition</i>] set append add unset echo <i>header</i> [<i>value</i>] [<i>early</i> env=[!] <i>variable</i>]
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_headers

HTTP

```
condition on success always (internal header) on success
" 2xx" always(" 2xx")
```

set
value

append
HTTP

add
() append

unset
() *value*

echo
headervalue

header() set, append, add, unset echo *header*

add, append, set *value value*(") *value value*

%%	(%)
----	-----

%t	(1970-1-1 00:00:00 UCT)"	t="
%D	" D="	
%{FOOBAR}e	FOOBAR	
%{FOOBAR}s	SSL FOOBAR(mod_ssl)	

```
"%s"Apache 2.1" %e"" SSLOptions +StdEnvVars"
" SSLOptions +StdEnvVars"" %e"" %s"
```

Header(early" ") env=..." (" env=!...") Header

[early](#) Header



RequestHeader

HTTP
RequestHeader set append add unset <i>header</i> [<i>value</i>] [<i>early</i> env=[!] <i>variable</i>]
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_headers
Apache 2.0

HTTP

set

append

HTTP

add

() append

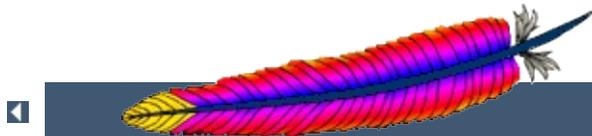
unset

() *value*

header() add, append, set *value* *value*(") unset
value *value* Header

RequestHeader(*early*" ") env=..." (" env=!...")
RequestHeader

[early](#) RequestHeaderApache



◀ **Apache HTTP Server 2.2**

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

| | | 2006126 |

Apache mod_ident

RFC1413ident
(E)
ident_module
mod_ident.c
Apache 2.1

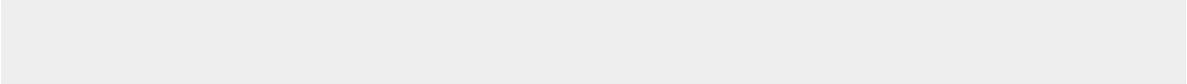
[RFC 1413](#)



Identity Check

RFC1413
IdentityCheck On Off
IdentityCheck Off
server config, virtual host, directory
(E)
mod_ident
Apache 2.1

```
identd RFC 1413(" %1")
```



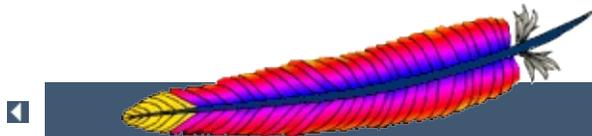
IdentityCheckTimeout



IdentityCheckTimeout

ident
IdentityCheckTimeout <i>seconds</i>
IdentityCheckTimeout 30
server config, virtual host, directory
(E)
mod_ident

ident"30"() [RFC 1413](#)



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_imagemap

(B)
imagemap_module
mod_imagemap.c

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 [AddHandlerSetHandler](#)) 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.



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`, `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 " ; .



```
#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>
```



ImapBase

Default base for imagemap files
ImapBase map referer URL
ImapBase http://servername/
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_imagemap

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/`.

- [UseCanonicalName](#)



ImapDefault

Default action when an imagemap is called with coordinates that are not explicitly mapped
ImapDefault error nocontent map referer URL
ImapDefault nocontent
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_imagemap

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.



ImapMenu

Action if no coordinates are given when calling an imagemap
ImapMenu none formatted semiformatted unformatted
server config, virtual host, directory, .htaccess
Indexes
(B)
mod_imagemap

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

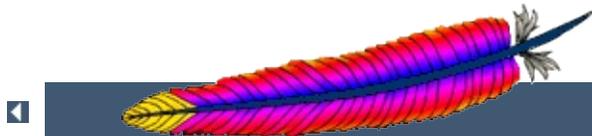
A formatted menu is the simplest menu. Comments in the imagemap file are ignored. A level one header is printed, then an hrule, then the links each on a separate line. The menu has a consistent, plain look close to that of a directory listing.

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.



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_include

(SSI)
(B)
include_module
mod_include.c
Implemented as an output filter since Apache 2.0

This module provides a filter which will process files before they are sent to the client. The processing is controlled by specially formatted SGML comments, referred to as *elements*. These elements allow conditional text, the inclusion of other files or programs, as well as the setting and printing of environment variables.



Server Side Includes are implemented by the INCLUDES [filter](#). If documents containing server-side include directives are given the extension .shtml, the following directives will make Apache parse them and assign the resulting document the mime type of text/html:

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

The following directive must be given for the directories containing the shtml files (typically in a [<Directory>](#) section, but this directive is also valid in .htaccess files if [AllowOverride Options](#) is set):

```
Options +Includes
```

For backwards compatibility, the server - parsed also activates the INCLUDES filter. As well, Apache will activate the INCLUDES filter for any document with mime type text/x-server-parsed-html text/x-server-parsed-html3 (and the resulting output will have the mime type text/html).

For more information, see our [Tutorial on Server Side Includes](#).



Files processed for server-side includes no longer accept requests with PATH_INFO (trailing pathname information) by default. You can use the [AcceptPathInfo](#) directive to configure the server to accept requests with PATH_INFO.



The document is parsed as an HTML document, with special commands embedded as SGML comments. A command has the syntax:

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

The value will often be enclosed in double quotes, but single quotes (') and backticks (`) are also possible. Many commands only allow a single attribute-value pair. Note that the comment terminator (- ->) should be preceded by whitespace to ensure that it isn't considered part of an SSI token. Note that the leading <! - -# is *one* token and may not contain any whitespaces.

The allowed elements are listed in the following table:

Element	Description
config	configure output formats
echo	print variables
exec	execute external programs
fsize	print size of a file
flastmod	print last modification time of a file
include	include a file
printenv	print all available variables
set	set a value of a variable

SSI elements may be defined by modules other than [mod_include](#). In fact, the [exec](#) element is provided by [mod_cgi](#), and will only be available if this module is loaded.

The config Element

This command controls various aspects of the parsing. The valid attributes are:

echormsg (Apache 2.1 and later)

The value is a message that is sent back to the client if the `echo` element attempts to echo an undefined variable. This overrides any `SSIUndefinedEcho` directives.

errormsg

The value is a message that is sent back to the client if an error occurs while parsing the document. This overrides any `SSIErrorMsg` directives.

sizefmt

The value sets the format to be used which displaying the size of a file. Valid values are `bytes` for a count in bytes, or `abbrev` for a count in Kb or Mb as appropriate, for example a size of 1024 bytes will be printed as "1K".

timefmt

The value is a string to be used by the `strftime(3)` library routine when printing dates.

The echo Element

This command prints one of the `include variables`, defined below. If the variable is unset, the result is determined by the `SSIUndefinedEcho` directive. Any dates printed are subject to the currently configured `timefmt`.

Attributes:

var

The value is the name of the variable to print.

encoding

Specifies how Apache should encode special characters contained in the variable before outputting them. If set to `none`,

no encoding will be done. If set to `url`, then URL encoding (also known as %-encoding; this is appropriate for use within URLs in links, etc.) will be performed. At the start of an `echo` element, the default is set to `entity`, resulting in entity encoding (which is appropriate in the context of a block-level HTML element, a paragraph of text). This can be changed by adding an `encoding` attribute, which will remain in effect until the next `encoding` attribute is encountered or the element ends, whichever comes first.

`encoding` attribute must *precede* the corresponding `var` attribute to be effective, and only special characters as defined in the ISO-8859-1 character encoding will be encoded. This encoding process may not have the desired result if a different character encoding is in use.

In order to avoid cross-site scripting issues, you should *always* encode user supplied data.

The `exec` Element

`exec` command executes a given shell command or CGI script. It requires `mod_cgi` to be present in the server. If `Options IncludesNOEXEC` is set, this command is completely disabled. The valid attributes are:

cgi

The value specifies a (%-encoded) URL-path to the CGI script. If the path does not begin with a slash (/), then it is taken to be relative to the current document. The document referenced by this path is invoked as a CGI script, even if the server would not normally recognize it as such. However, the directory containing the script must be enabled for CGI scripts (with `ScriptAlias Options ExecCGI`).

The CGI script is given the `PATH_INFO` and query string (`QUERY_STRING`) of the original request from the client; these *cannot* be specified in the URL path. The include variables will be available to the script in addition to the standard [CGI](#) environment.

```
<!--#exec cgi="/cgi-bin/example.cgi" -->
```

If the script returns a `Location:` header instead of output, then this will be translated into an HTML anchor.

[include virtual](#) element should be used in preference to `exec cgi`. In particular, if you need to pass additional arguments to a CGI program, using the query string, this cannot be done with `exec cgi`, but can be done with `include virtual`, as shown here:

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value" -->
```

cmd

The server will execute the given string using `/bin/sh`. The [include variables](#) are available to the command, in addition to the usual set of CGI variables.

The use of [#include virtual](#) is almost always preferred to using either `#exec cgi#exec cmd`. The former (`#include virtual`) uses the standard Apache sub-request mechanism to include files or scripts. It is much better tested and maintained.

In addition, on some platforms, like Win32, and on unix when using [suexec](#), you cannot pass arguments to a command in an `exec` directive, or otherwise include spaces in the command.

Thus, while the following will work under a non-suexec configuration on unix, it will not produce the desired result under Win32, or when running suexec:

```
<!--#exec cmd="perl /path/to/perlscript arg1  
arg2" -->
```

The **fsize** Element

This command prints the size of the specified file, subject to the `sizefmt` format specification. Attributes:

file

The value is a path relative to the directory containing the current document being parsed.

virtual

The value is a (%-encoded) URL-path. If it does not begin with a slash (/) then it is taken to be relative to the current document. Note, that this does *not* print the size of any CGI output, but the size of the CGI script itself.

The **flastmod** Element

This command prints the last modification date of the specified file, subject to the `timefmt` format specification. The attributes are the same as for the [fsize](#) command.

The **include** Element

This command inserts the text of another document or file into the parsed file. Any included file is subject to the usual access control. If the directory containing the parsed file has [Options](#) `IncludesNOEXEC` set, then only documents with a text [MIME-type](#) (`text/plain`, `text/html` etc.) will be included. Otherwise CGI scripts are invoked as normal using the complete URL given in the

command, including any query string.

An attribute defines the location of the document; the inclusion is done for each attribute given to the include command. The valid attributes are:

file

The value is a path relative to the directory containing the current document being parsed. It cannot contain `../`, nor can it be an absolute path. Therefore, you cannot include files that are outside of the document root, or above the current document in the directory structure. The `virtual` attribute should always be used in preference to this one.

virtual

The value is a (%-encoded) URL-path. The URL cannot contain a scheme or hostname, only a path and an optional query string. If it does not begin with a slash (/) then it is taken to be relative to the current document.

A URL is constructed from the attribute, and the output the server would return if the URL were accessed by the client is included in the parsed output. Thus included files can be nested.

If the specified URL is a CGI program, the program will be executed and its output inserted in place of the directive in the parsed file. You may include a query string in a CGI url:

```
<!--#include virtual="/cgi-bin/example.cgi?  
argument=value" -->
```

`include virtual` should be used in preference to `exec cgi` to include the output of CGI programs into an HTML document.

The printenv Element

This prints out a listing of all existing variables and their values. Special characters are entity encoded (see the [echo](#) element for details) before being output. There are no attributes.

```
<!--#printenv -->
```

The set Element

This sets the value of a variable. Attributes:

var

The name of the variable to set.

value

The value to give a variable.

```
<!--#set var="category" value="help" -->
```



In addition to the variables in the standard CGI environment, these are available for the `echo` command, for `ifelif`, and to any program invoked by the document.

DATE_GMT

The current date in Greenwich Mean Time.

DATE_LOCAL

The current date in the local time zone.

DOCUMENT_NAME

The filename (excluding directories) of the document requested by the user.

DOCUMENT_URI

The (%-decoded) URL path of the document requested by the user. Note that in the case of nested include files, this is *not* the URL for the current document.

LAST_MODIFIED

The last modification date of the document requested by the user.

QUERY_STRING_UNESCAPED

If a query string is present, this variable contains the (%-decoded) query string, which is *escaped* for shell usage (special characters like `&` etc. are preceded by backslashes).



Variable Substitution

Variable substitution is done within quoted strings in most cases where they may reasonably occur as an argument to an SSI directive. This includes the `config`, `exec`, `flastmod`, `fsize`, `include`, `echo`, and `set` directives, as well as the arguments to conditional operators. You can insert a literal dollar sign into the string using backslash quoting:

```
<!--#if expr="$a = \$test" -->
```

If a variable reference needs to be substituted in the middle of a character sequence that might otherwise be considered a valid identifier in its own right, it can be disambiguated by enclosing the reference in braces, *a /a* shell substitution:

```
<!--#set var="Zed"  
value="{REMOTE_HOST}_{REQUEST_METHOD}" -->
```

This will result in the Zed variable being set to "X_Y" if `REMOTE_HOST` is "X" and `REQUEST_METHOD` is "Y".

The below example will print "in foo" if the `DOCUMENT_URI` is `/foo/file.html`, "in bar" if it is `/bar/file.html` and "in neither" otherwise:

```
<!--#if expr="$DOCUMENT_URI" = "/foo/file.html"  
-->  
    in foo  
<!--#elif expr="$DOCUMENT_URI" =  
"/bar/file.html" -->  
    in bar  
<!--#else -->  
    in neither  
<!--#endif -->
```



The basic flow control elements are:

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

`if` element works like an `if` statement in a programming language. The test condition is evaluated and if the result is true, then the text until the next `elif`, `elseif` element is included in the output stream.

`elseif` statements are be used to put text into the output stream if the original `test_condition` was false. These elements are optional.

`endif` element ends the `if` element and is required.

`test_condition` is one of the following:

string

true if *string* is not empty

string1 = string2

string1 == string2

string1 != string2

Compare *string1* with *string2*. If *string2* has the form `/string2/` then it is treated as a regular expression. Regular expressions are implemented by the [PCRE](#) engine and have the same syntax as those in [perl 5](#). Note that `==` is just an alias for `=` and behaves exactly the same way.

If you are matching positive (`==`), you can capture grouped parts of the regular expression. The captured parts are stored in the special variables `$1 .. $9`.

```
<!--#if expr="$QUERY_STRING = /^sid=([a-zA-Z0-9]+)/" -->
  <!--#set var="session" value="$1" -->
<!--#endif -->
```

string1 < string2
string1 <= string2
string1 > string2
string1 >= string2

Compare *string1* with *string2*. Note, that strings are compared *literally* (using `strcmp(3)`). Therefore the string "100" is less than "20".

(test_condition)
true if *test_condition* is true

! test_condition
true if *test_condition* is false

test_condition1 && test_condition2
true if both *test_condition1**test_condition2* are true

test_condition1 || test_condition2
true if either *test_condition1**test_condition2* is true

"=" and "!=" bind more tightly than "&&" and "||". "!" binds most tightly. Thus, the following are equivalent:

```
<!--#if expr="$a = test1 && $b = test2" -->
<!--#if expr="($a = test1) && ($b = test2)" -->
```

The boolean operators `&& ||` share the same priority. So if you want to bind such an operator more tightly, you should use parentheses.

Anything that's not recognized as a variable or an operator is treated as a string. Strings can also be quoted: `'string'`. Unquoted strings

can't contain whitespace (blanks and tabs) because it is used to separate tokens such as variables. If multiple strings are found in a row, they are concatenated using blanks. So,

```
string1 string2 results in string1 string2
```

```
'string1 string2' results in string1 string2.
```

Optimization of Boolean Expressions

If the expressions become more complex and slow down processing significantly, you can try to optimize them according to the evaluation rules:

- Expressions are evaluated from left to right
- Binary boolean operators (&& | |) are short circuited wherever possible. In conclusion with the rule above that means, [mod_include](#) evaluates at first the left expression. If the left result is sufficient to determine the end result, processing stops here. Otherwise it evaluates the right side and computes the end result from both left and right results.
- Short circuit evaluation is turned off as long as there are regular expressions to deal with. These must be evaluated to fill in the backreference variables (\$1 .. \$9).

If you want to look how a particular expression is handled, you can recompile [mod_include](#) using the `-DDEBUG_INCLUDE` compiler option. This inserts for every parsed expression tokenizer information, the parse tree and how it is evaluated into the output sent to the client.



SSIEndTag

String that ends an include element
SSIEndTag <i>tag</i>
SSIEndTag "-->"
server config, virtual host
(B)
mod_include
Apache 2.0.30

This directive changes the string that [mod_include](#) looks for to mark the end of an include element.

```
SSIEndTag "%>"
```

- [SSISStartTag](#)



Error message displayed when there is an SSI error
SSIErrorMsg <i>message</i>
SSIErrorMsg "[an error occurred while processing this directive]"
server config, virtual host, directory, .htaccess
All
(B)
mod_include
Apache 2.0.30

SSIErrorMsg directive changes the error message displayed when mod_include encounters an error. For production servers you may consider changing the default error message to "`<!-- Error -->`" so that the message is not presented to the user.

This directive has the same effect as the `<!--#config errmsg=message -->` element.

```
SSIErrorMsg "<!-- Error -->"
```



SSIStartTag

String that starts an include element
SSIStartTag <i>tag</i>
SSIStartTag "<! - -#"
server config, virtual host
(B)
mod_include
Apache 2.0.30

This directive changes the string that [mod_include](#) looks for to mark an include element to process.

You may want to use this option if you have 2 servers parsing the output of a file each processing different commands (possibly at different times).

```
SSIStartTag "<%"  
SSIEndTag "%>"
```

The example given above, which also specifies a matching [SSIEndTag](#), will allow you to use SSI directives as shown in the example below:

SSI directives with alternate start and end tags

```
<%printenv %>
```

- [SSIEndTag](#)



Configures the format in which date strings are displayed
<code>SSITimeFormat <i>formatstring</i></code>
<code>SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"</code>
server config, virtual host, directory, .htaccess
All
(B)
mod_include
Apache 2.0.30

This directive changes the format in which date strings are displayed when echoing DATE environment variables. The *formatstring* is as in `strftime(3)` from the C standard library.

This directive has the same effect as the `<!--#config timefmt=formatstring -->` element.

```
SSITimeFormat "%R, %B %d, %Y"
```

The above directive would cause times to be displayed in the format "22:26, June 14, 2002".



String displayed when an unset variable is echoed
SSIUndefinedEcho <i>string</i>
SSIUndefinedEcho "(none)"
server config, virtual host, directory, .htaccess
All
(B)
mod_include
Apache 2.0.34

This directive changes the string that [mod_include](#) displays when a variable is not set and "echoed".

```
SSIUndefinedEcho "<!-- undef -->"
```



Parse SSI directives in files with the execute bit set
XBitHack on off full
XBitHack off
server config, virtual host, directory, .htaccess
Options
(B)
mod_include

XBitHack directive controls the parsing of ordinary html documents. This directive only affects files associated with the [MIME-type](#) text/html. **XBitHack** can take on the following values:

off

No special treatment of executable files.

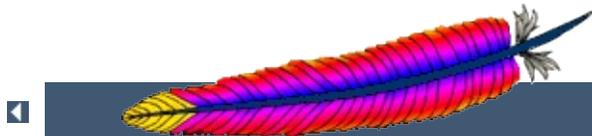
on

Any text/html file that has the user-execute bit set will be treated as a server-parsed html document.

full

As for on but also test the group-execute bit. If it is set, then set the Last-modified date of the returned file to be the last modified time of the file. If it is not set, then no last-modified date is sent. Setting this bit allows clients and proxies to cache the result of the request.

You would not want to use the full option, unless you assure the group-execute bit is unset for every SSI script which might #include a CGI or otherwise produces different output on each hit (or could potentially change on subsequent requests).



Apache HTTP Server 2.2

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

| | | 2006321 |

Apache mod_info

ApacheWeb
(E)
info_module
mod_info.c

mod_infohttpd.conf

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

<Location>mod_authz_host

```
<Location /server-info>  
  SetHandler server-info  
  Order deny,allow  
  Deny from all  
  Allow from yourcompany.com  
</Location>
```

http://your.host.example.com/server-info



mod_info .htaccess

/

mod_authz_host

```
<Location /server-info>
  SetHandler server-info
  Order allow,deny
  #
  Allow from 127.0.0.1
  #
  Allow from 192.168.1.17
</Location>
```



server-info <http://your.host.example.com/server-info?config>

?<module-name>

?config

?hooks

(Hook)

?list

?server



mod_info

- ServerRoot, LoadModule, LoadFile
- Include, <IfModule>, <IfDefine> Include
-
- .htaccess
- mod_info</Directory>
- (mod_ssl)

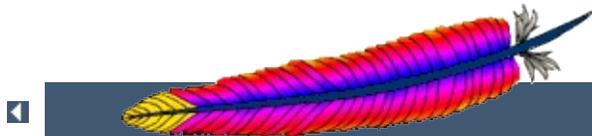


server-info
AddModuleInfo <i>module-name string</i>
server config, virtual host
(E)
mod_info
Apache 1.3

*stringmodule-name*HTML

```
AddModuleInfo mod_deflate.c 'See <a \
  href="http://www.apache.org/docs/2.2/mod/mod_deflat
  http://www.apache.org/docs/2.2/mod/mod_deflate.html
```

||||



◀ **Apache HTTP Server 2.2**

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

| | | 2006126 |

Apache mod_isapi

WindowsISAPI
(B)
isapi_module
mod_isapi.c
Win32

(Internet Server extension API)WindowsApache(ISAPI)

ISAPI(.dll)ApacheISAPIISAPI

Apache



[AddHandler](#)isapi-isaISAPI.dllISAPIIhttpd.conf

```
AddHandler isapi-isa .dll
```

Apachehttpd.confApache

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

ISAPIISAPICGIISAPI"

[Options](#) ExecCGI"

[mod_isapi](#)ISAPI



ApacheISAPI/O(Microsoft-specific)ISAPI 2.0Apache/OISAPI
ISAPI/O"

IISISAPIApacheISAPI
ApacheISAPIApache

[ISAPICacheFile](#)

ApacheISAPI **ISAPIISAPI**



Apache 2.0 [mod_isapi](#) ServerSupportFunction

HSE_REQ_SEND_URL_REDIRECT_RESP

URL(http://server/location)

HSE_REQ_SEND_URL

URL(/location)

HSE_REQ_SEND_URLApache

HSE_REQ_SEND_RESPONSE_HEADER

()ApacheNULLNULL

HSE_REQ_DONE_WITH_SESSION

ApacheISAPI

HSE_REQ_MAP_URL_TO_PATH

Apache

HSE_APPEND_LOG_PARAMETER

- [CustomLog](#) \"%{isapi-parameter}n\"
- " [ISAPIAppendLogToQuery](#) On" %q"
- " [ISAPIAppendLogToErrors](#) On"

{isapi-parameter}n

HSE_REQ_IS_KEEP_CONN

Keep-Alive

HSE_REQ_SEND_RESPONSE_HEADER_EX

fKeepConn

HSE_REQ_IS_CONNECTED

ServerSupportFunctionApache FALSE GetLastError
ERROR_INVALID_PARAMETER

ReadClient(ISAPIReadAheadBuffer)
ISAPIReadAheadBuffer(ISAPI)ISAPIISAPI
ReadClient

WriteClient HSE_IO_SYNC("0") WriteClient FALSE
GetLastErrorERROR_INVALID_PARAMETER

GetServerVariable () ALL_HTTPALL_RAW Apache CGI
GetServerVariable

Apache 2.0 mod_isapiISAPI/I/O TransmitFileApacheISAPI
.dlls Apache1.3 mod_isapi



ISAPIHSE_APPEND_LOG_PARAMETER

ISAPIHSE_APPEND_LOG_PARAMETER
ISAPIAppendLogToErrors on off
ISAPIAppendLogToErrors off
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_isapi

ISAPIHSE_APPEND_LOG_PARAMETER



ISAPIHSE_APPEND_LOG_PARAMETER

ISAPIHSE_APPEND_LOG_PARAMETER
ISAPIAppendLogToQuery on off
ISAPIAppendLogToQuery on
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_isapi

ISAPIHSE_APPEND_LOG_PARAMETER([CustomLog](#) %q)



ISAPI
ISAPICacheFile <i>file-path</i> [<i>file-path</i>] ...
server config, virtual host
(B)
mod_isapi

ApacheISAPI

ServerRoot



ISAPI FakeAsync

ISAPI
ISAPIFakeAsync on off
ISAPIFakeAsync off
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_isapi

onISAPI



ISAPI Log Not Supported

ISAPI
ISAPILogNotSupported on off
ISAPILogNotSupported off
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_isapi

ISAPIonISAPIOff

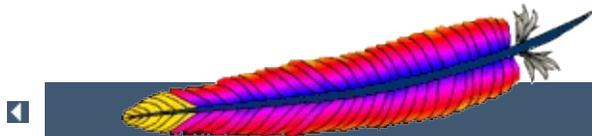


ISAPI ReadAheadBuffer

ISAPI
ISAPIReadAheadBuffer <i>size</i>
ISAPIReadAheadBuffer 49152
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_isapi

ISAPI ReadClientISAPI ReadClientISAPI





Apache HTTP Server 2.2

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

| | | ??? |

Apache mod_ldap

LDAPLDAP
(E)
ldap_module
util_ldap.c
Apache 2.0.41

LDAPLDAPLDAPLDAP

LDAPAPUApache `configure` --with-ldap

SSL/TLS [APRLDAP SDK](#) [OpenLDAP SDK\(2.x\)](#), [Novell LDAP SDK](#), [Mozilla LDAP SDK](#), Solaris LDAP SDK (Mozilla), Microsoft LDAP SDK, [iPlanet \(Netscape\) SDK](#) [APR](#)



mod_ldapmod_authnz_ldapHTTP

```
# LDAP
# LDAPmod_ldapmod_authnz_ldap
# "yourdomain.example.com"

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>
```



LDAPLDAPunbind->connect->rebindHTTPKeep-Alive

LDAPLDAPApache

ApacheLDAP



[mod_ldap](#)LDAPApache[mod_authnz_ldap](#)LDAP

[mod_ldap](#)LDAPsearch/bind *search/bindcompare* *operation*
LDAP URL

Search/Bind

LDAPSearch/bind()

[mod_ldap](#)DN [mod_ldap](#) [mod_ldap](#)
search/bind

[LDAPCacheEntries](#)[LDAPCacheTTL](#)

Operation

[mod_ldap](#)LDAP

[LDAPOpCacheEntries](#)[LDAPOpCacheTTL](#)

[mod_ldap](#) ldap-status [mod_ldap](#)

```
<Location /server/cache-info>  
  SetHandler ldap-status  
</Location>
```

URL <http://servername/cache-info> [mod_ldap](#)
Apache [httpdURL](#) [httpd](#)



[LDAPTrustedGlobalCert](#), [LDAPTrustedClientCert](#),
[LDAPTrustedModeLDAPSSL/TSLCA](#)(none, SSL,
TLS/STARTTLS)

```
# 636SSL LDAPmod_ldapmod_authnz_ldap  
# "yourdomain.example.com"
```

```
LDAPTrustedGlobalCert CA_DER /certs/certfile.der
```

```
<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>
```

```
# 389TLS LDAPmod_ldapmod_authnz_ldap  
# "yourdomain.example.com"
```

```
LDAPTrustedGlobalCert CA_DER /certs/certfile.der
```

```
<Location /ldap-status>  
  SetHandler ldap-status  
  Order deny,allow  
  Deny from all  
  Allow from yourdomain.example.com  
  AuthLDAPEnabled on  
  LDAPTrustedMode TLS AuthLDAPURL  
  ldap://127.0.0.1/dc=example,dc=com?uid?one  
  AuthLDAPAuthoritative on
```

```
    require valid-user
</Location>
```



The different LDAP SDKs have widely different methods of setting and handling both CA and client side certificates.

If you intend to use SSL or TLS, read this section CAREFULLY so as to understand the differences between configurations on the different LDAP toolkits supported.

Netscape/Mozilla/iPlanet SDK

CA certificates are specified within a file called cert7.db. The SDK will not talk to any LDAP server whose certificate was not signed by a CA specified in this file. If client certificates are required, an optional key3.db file may be specified with an optional password. The secmod file can be specified if required. These files are in the same format as used by the Netscape Communicator or Mozilla web browsers. The easiest way to obtain these files is to grab them from your browser installation.

Client certificates are specified per connection using the LDAPTrustedClientCert directive by referring to the certificate "nickname". An optional password may be specified to unlock the certificate's private key.

The SDK supports SSL only. An attempt to use STARTTLS will cause an error when an attempt is made to contact the LDAP server at runtime.

```
# Specify a Netscape CA certificate file
LDAPTrustedGlobalCert CA_CERT7_DB /certs/cert7.db
# Specify an optional key3.db file for client
certificate support
LDAPTrustedGlobalCert CERT_KEY3_DB /certs/key3.db
# Specify the secmod file if required
LDAPTrustedGlobalCert CA_SECMOD /certs/secmod
<Location /ldap-status>
```

```
SetHandler ldap-status
Order deny,allow
Deny from all
Allow from yourdomain.example.com
AuthLDAPEnabled on
LDAPTrustedClientCert CERT_NICKNAME <nickname>
[password]
AuthLDAPURL
ldaps://127.0.0.1/dc=example,dc=com?uid?one
AuthLDAPAuthoritative on
require valid-user
</Location>
```

Novell SDK

One or more CA certificates must be specified for the Novell SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.

Note: Client certificates are specified globally rather than per connection, and so must be specified with the LDAPTrustedGlobalCert directive as below. Trying to set client certificates via the LDAPTrustedClientCert directive will cause an error to be logged when an attempt is made to connect to the LDAP server..

The SDK supports both SSL and STARTTLS, set using the LDAPTrustedMode parameter. If an ldaps:// URL is specified, SSL mode is forced, override this directive.

```
# Specify two CA certificate files
LDAPTrustedGlobalCert CA_DER /certs/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
# Specify a client certificate file and key
LDAPTrustedGlobalCert CERT_BASE64 /certs/cert1.pem
LDAPTrustedGlobalCert KEY_BASE64 /certs/key1.pem
```

```
[password]
# Do not use this directive, as it will throw an
error
#LDAPTrustedClientCert CERT_BASE64
/certs/cert1.pem
```

OpenLDAP SDK

One or more CA certificates must be specified for the OpenLDAP SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.

Client certificates are specified per connection using the LDAPTrustedClientCert directive.

The documentation for the SDK claims to support both SSL and STARTTLS, however STARTTLS does not seem to work on all versions of the SDK. The SSL/TLS mode can be set using the LDAPTrustedMode parameter. If an ldaps:// URL is specified, SSL mode is forced. The OpenLDAP documentation notes that SSL (ldaps://) support has been deprecated to be replaced with TLS, although the SSL functionality still works.

```
# Specify two CA certificate files
LDAPTrustedGlobalCert CA_DER /certs/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPEnabled on
  LDAPTrustedClientCert CERT_BASE64
  /certs/cert1.pem
  LDAPTrustedClientCert KEY_BASE64
  /certs/key1.pem
```

```
AuthLDAPURL
ldaps://127.0.0.1/dc=example,dc=com?uid?one
AuthLDAPAuthoritative on
require valid-user
</Location>
```

Solaris SDK

SSL/TLS for the native Solaris LDAP libraries is not yet supported. If required, install and use the OpenLDAP libraries instead.

Microsoft SDK

SSL/TLS certificate configuration for the native Microsoft LDAP libraries is done inside the system registry, and no configuration directives are required.

Both SSL and TLS are supported by using the ldaps:// URL format, or by using the LDAPTrustedMode directive accordingly.

Note: The status of support for client certificates is not yet known for this toolkit.



LDAP Cache Entries

LDAP
LDAPCacheEntries <i>number</i>
LDAPCacheEntries 1024
server config
(E)
mod_ldap

LDAPsearch/bind0search/bind1024



LDAP Cache TTL

search/bind
LDAPCacheTTL <i>seconds</i>
LDAPCacheTTL 600
server config
(E)
mod_ldap

search/bind600(10)



LDAPConnectionTimeout

LDAPConnectionTimeout <i>seconds</i>
server config
(E)
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.



LDAP OpCache Entries

LDAP compare
LDAPOpCacheEntries <i>number</i>
LDAPOpCacheEntries 1024
server config
(E)
mod_ldap

[mod_ldap](#)LDAP compare10240



LDAPOpCacheTTL

LDAPOpCacheTTL <i>seconds</i>
LDAPOpCacheTTL 600
server config
(E)
mod_ldap

600



LDAPSharedCacheFile <i>directory-path/filename</i>
server config
(E)
mod_ldap

()



LDAPSharedCacheSize <i>bytes</i>
LDAPSharedCacheSize 102400
server config
(E)
mod_ldap

Byte100KB



Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates.
LDAPTrustedClientCert <i>type directory-path/filename/nickname [password]</i>
server config, virtual host, directory, .htaccess
(E)
mod_ldap

It specifies the directory path, file name or nickname of a per connection client certificate used when establishing an SSL or TLS connection to an LDAP server. Different locations or directories may have their own independant client certificate settings. Some LDAP toolkits (notably Novell) do not support per connection client certificates, and will throw an error on LDAP server connection if you try to use this directive (Use the LDAPTrustedGlobalCert directive instead for Novell client certificates - See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- CERT_DER - binary DER encoded client certificate
- CERT_BASE64 - PEM encoded client certificate
- CERT_NICKNAME - Client certificate "nickname" (Netscape SDK)
- KEY_DER - binary DER encoded private key
- KEY_BASE64 - PEM encoded private key



Sets the file or database containing global trusted Certificate Authority or global client certificates
<code>LDAPTrustedGlobalCert type directory-path/filename [password]</code>
server config
(E)
mod_ldap

It specifies the directory path and file name of the trusted CA certificates and/or system wide client certificates `mod_ldap` should use when establishing an SSL or TLS connection to an LDAP server. Note that all certificate information specified using this directive is applied globally to the entire server installation. Some LDAP toolkits (notably Novell) require all client certificates to be set globally using this directive. Most other toolkits require clients certificates to be set per Directory or per Location using `LDAPTrustedClientCert`. If you get this wrong, an error may be logged when an attempt is made to contact the LDAP server, or the connection may silently fail (See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- `CA_DER` - binary DER encoded CA certificate
- `CA_BASE64` - PEM encoded CA certificate
- `CA_CERT7_DB` - Netscape cert7.db CA certificate database file
- `CA_SECMOD` - Netscape secmod database file
- `CERT_DER` - binary DER encoded client certificate
- `CERT_BASE64` - PEM encoded client certificate
- `CERT_KEY3_DB` - Netscape key3.db client certificate database file
- `CERT_NICKNAME` - Client certificate "nickname" (Netscape SDK)
- `CERT_PFX` - PKCS#12 encoded client certificate (Novell SDK)

- KEY_DER - binary DER encoded private key
- KEY_BASE64 - PEM encoded private key
- KEY_PFX - PKCS#12 encoded private key (Novell SDK)



LDAPTrustedMode

Specifies the SSL/TLS mode to be used when connecting to an LDAP server.
LDAPTrustedMode <i>type</i>
server config, virtual host, directory, .htaccess
(E)
mod_ldap

The following modes are supported:

- NONE - no encryption
- SSL - ldaps:// encryption on default port 636
- TLS - STARTTLS encryption on default port 389

Not all LDAP toolkits support all the above modes. An error message will be logged at runtime if a mode is not supported, and the connection to the LDAP server will fail.

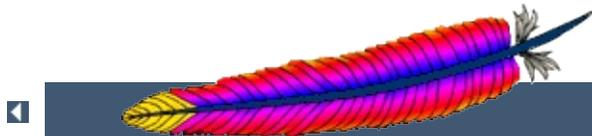
If an ldaps:// URL is specified, the mode becomes SSL and the setting of LDAPTrustedMode is ignored.



LDAPVerifyServerCert

Force server certificate verification
LDAPVerifyServerCert <i>On Off</i>
LDAPVerifyServerCert On
server config
(E)
mod_ldap

Specifies whether to force the verification of a server certificate when establishing an SSL connection to the LDAP server.



Apache HTTP Server 2.2

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

| | | 2006126 |

Apache mod_log_config

(B)
log_config_module
mod_log_config.c

TransferLog LogFormat CustomLog
TransferLogCustomLog



LogFormatCustomLogC"\n""\t""\\"

" %" "

%%	(Apache2.0.44)
%a	IP
%A	IP
%B	HTTP
%b	CLFHTTP' -'0
% {Foobar}C	cookieFoobar
%D	
% {FOOBAR}e	FOOBAR
%f	
%h	
%H	
% {Foobar}i	Foobar :
%l	(identd) IdentityCheck " On""-"
%m	
% {Foobar}n	Foobar
% {Foobar}o	Foobar :
%p	
%P	PID
% {format}P	PIDTID(ID) <i>format</i> pidtid(2.0.46)hexid(APR1.2.0)

%q	(" ?")
%r	
%s	--- %>s
%t	()
% {format}t	strftime(3)()
%T	
%u	(status(%s)401)
%U	URL
%v	<u>ServerName</u>
%V	<u>UseCanonicalName</u>
%X	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> X= += -= </div> (1.3 %cSSL %{var}c)
%I	<u>mod_logio</u>
%O	<u>mod_logio</u>

```
"%" %400,501{User-agent}i"400501 User-agent
" -" !" %!200,304,302{Referer}i" 200,304,302
Referer
```

```
"<">" %s, %U, %T, %D, %r %>s %<u
```

```
2.0.46      %r, %i, %o (")(\    \"    \\ C(    \n, \t )
\xhh (hh16)2.0.46
```

```
2.0(1.3)   %b   %B HTTP(SSL)      mod_logio %0
```

(CLF)

```
"%h %l %u %t \"%r\" %>s %b"
```

```
"%v %h %l %u %t \"%r\" %>s %b"
```

NCSA/

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
```

Referer

```
"%{Referer}i -> %U"
```

Agent(Browser)

```
"%{User-agent}i"
```



Apache



BufferedLogs

BufferedLogs On Off
BufferedLogs Off
server config
(B)
mod_log_config
Apache 2.0.41

BufferedLogs mod_log_config



CookieLog

cookies
CookieLog <i>filename</i>
server config, virtual host
(B)
mod_log_config

CookieLogcookies ServerRoot mod_cookies



<code>CustomLog file pipe format nickname [env=[!]environment-variable]</code>
server config, virtual host
(B)
mod_log_config

CustomLog

file

ServerRoot

pipe

" |"

[httpdhttpdroot root](#)

UNIX(\)(/)(/)

LogFormatnickname format

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

#
```

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

```
(" env=!name")
```

mod_setenvif/ mod_rewriteGIF

```
SetEnvIf Request_URI \.gif$ gif-image  
CustomLog gif-requests.log common env=gif-image  
CustomLog nongif-requests.log common env=!gif-  
image
```

RefererIgnore

```
SetEnvIf Referer example\.com localreferer  
CustomLog referer.log referer env=!localreferer
```



LogFormat

LogFormat <i>format</i> [<i>nickname</i>]
LogFormat "%h %l %u %t \"%r\" %>s %b"
server config, virtual host
(B)
mod_log_config

LogFormat TransferLog *format* *nickname*
LogFormat

LogFormat *format**nickname* LogFormatCustomLog
LogFormat*nickname* TransferLog
 LogFormat(%)

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b"  
vhost_common
```



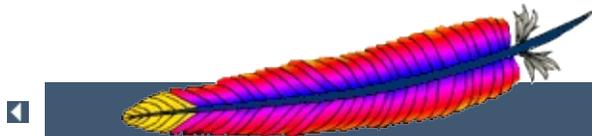
TransferLog

TransferLog <i>file pipe</i>
server config, virtual host
(B)
mod_log_config

[CustomLog](#)

[LogFormat](#)

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



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_log_forensic

""
(E)
log_forensic_module
mod_log_forensic.c
<u>mod_unique_id</u> is no longer required since version 2.1

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).

check_forensic script, which can be found in the distribution's support directory, may be helpful in evaluating the forensic log output.



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
```

`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

Sets filename of the forensic log
ForensicLog <i>filename pipe</i>
server config, virtual host
(E)
mod_log_forensic

ForensicLog directive is used to log requests to the server for forensic analysis. Each log entry is assigned a unique ID which can be associated with the request using the normal **CustomLog** directive. **mod_log_forensic** creates a token called `forensic-id`, which can be added to the transfer log using the `%{forensic-id}n` format string.

The argument, which specifies the location to which the logs will be written, can take one of the following two types of values:

filename

A filename, relative to the **ServerRoot**.

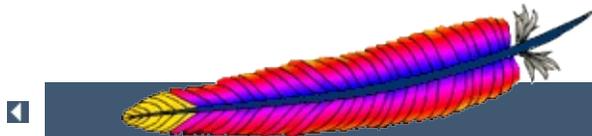
pipe

The pipe character "|", followed by the path to a program to receive the log information on its standard input. The program name can be specified relative to the **ServerRoot** directive.

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.

When entering a file path on non-Unix platforms, care should

be taken to make sure that only forward slashes are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.



◀ Apache HTTP Server 2.2

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

| | | 2006126 |

Apache mod_logio

/HTTP
(E)
logio_module
mod_logio.c

/SSL/TLSSSL/TLS

[mod_log_config](#)



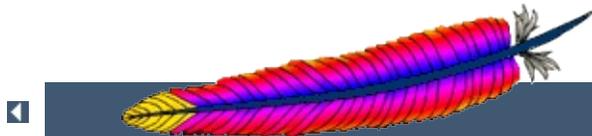
" %"

%I	
%O	

I/O

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\" %I %O"
```

||||



◀ **Apache HTTP Server 2.2**

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

| | | 2006126 |

Apache mod_mem_cache

(E)
mem_cache_module
mod_mem_cache.c

[mod_cache](#) [mod_cache](#) [mod_mem_cache](#)
[mod_mem_cache](#) [mod_proxyProxyPass\(\)](#)

URI



MCacheMaxObjectCount

MCacheMaxObjectCount <i>value</i>
MCacheMaxObjectCount 1009
server config
(E)
mod_mem_cache

MCacheMaxObjectCount
MCacheRemovalAlgorithm

MCacheMaxObjectCount 13001



MCacheMaxObjectSize

()
MCacheMaxObjectSize <i>bytes</i>
MCacheMaxObjectSize 10000
server config
(E)
mod_mem_cache

MCacheMaxObjectSize(Byte)

MCacheMaxObjectSize 6400000

MCacheMaxObjectSize[MCacheMinObjectSize](#)



MCacheMaxStreamingBuffer

MCacheMaxStreamingBuffer <i>size_in_bytes</i>
MCacheMaxStreamingBuffer 100000MCacheMaxObjectSize
server config
(E)
mod_mem_cache

MCacheMaxStreamingBuffer Content-LengthCGI
Content-Length MCacheMaxStrea
Content-Length

MCacheMaxStreamingBuffer mod_mem_cache

64KB
MCacheMaxStreamingBuffer 65536



MCacheMinObjectSize

()
MCacheMinObjectSize <i>bytes</i>
MCacheMinObjectSize 0
server config
(E)
mod_mem_cache

MCacheMinObjectSize

```
MCacheMinObjectSize 10000
```



MCacheRemovalAlgorithm

MCacheRemovalAlgorithm LRU GDSF
MCacheRemovalAlgorithm GDSF
server config
(E)
mod_mem_cache

MCacheRemovalAlgorithm

LRU ()

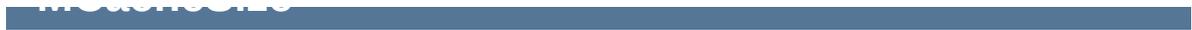
LRU

GDSF (GreedyDual-Size)

GDSF

MCacheRemovalAlgorithm GDSF
MCacheRemovalAlgorithm LRU





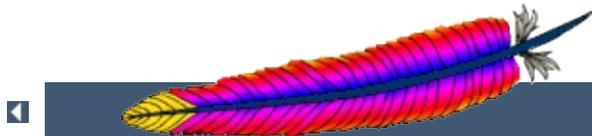
KB
MCacheSize <i>KBytes</i>
MCacheSize 100
server config
(E)
mod_mem_cache

MCacheSizeKB(1024-byte)

MCacheRemovalAlgorithm

MCacheSize 700000

MCacheSizeMCacheMaxObjectSize



◀ **Apache HTTP Server 2.2**

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

| | | 2006127 |

Apache mod_mime

(/)(MIME//)
(B)
mime_module
mod_mime.c

""MIME

mod_negotiation

AddCharset, AddEncoding, AddLanguage, AddType
MIME() TypesConfigMIME

mod_mime AddHandler, AddOutputFilter,
AddInputFilter MultiviewsMatchmod_negotiation
Multiview

mod_mime core(<Location>, <Directory>, <Files>)
 ForceType, SetHandler, SetInputFilter,
SetOutputFilter coremod_mime

Last-Modified()""()



welcome.html.frtext/html welcome.fr.html
welcome.gif.htmlMIMEtext/html .gif!

welcome.html.en.deContent-Language: en, de
Content-Type: text/html

MIME .imap(mod_imagemap) imap-file
.htmlMIMEtext/html world.imap.htmlimap-file
text/htmlMIME imap-file mod_imagemap



MIME gzip pgp UUencodingUUencoding
ASCII()

[HTTP/1.1 RFC14.11](#)

*"Content-Encoding""Content-Type""Content-
Encoding"*

()

Microsoft Word .doc Microsoft Word .zip pkzip
Resume .doc .zip pkzip Word

ApacheContent-encoding

Content-encoding: pkzip



HTTP

([mod_negotiation](#)) [AddCharset](#),
[AddEncoding](#), [AddLanguage](#), [AddType\(MimeMagicFile\)](#)
[AddHandler](#), [AddInputFilter](#), [AddOutputFilter](#)
[MultiviewsMatch](#)

Apache Content-Language Content-Type

```
Content-Language: en, fr  
Content-Type: text/plain; charset=ISO-8859-1
```

charset



AddCharset

AddCharset <i>charset extension [extension] ...</i>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

AddCharset *charsetextension*[MIME](#) *extension*

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

```
xxxx.ja.jisISO-2022-JP( xxxx.jis.ja) AddCharset
()
```

extension

- [mod_negotiation](#)
- [AddDefaultCharset](#)



AddEncoding

AddEncoding <i>MIME-enc extension</i> [<i>extension</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

AddEncoding *extension* *MIME-enc* *extension*

```
AddEncoding x-gzip .gz
AddEncoding x-compress .Z
```

.gz x-gzip .Z x-compress

x-gzip x-compress gzip compress Apache" x-"
Apache(x-foofoo)Apa
x-gzip x-compress deflate" x-"

extension



ADDHANDLER

AddHandler <i>handler-name extension [extension] ...</i>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

extension[handler-name](#) *extension* .cgiCGI

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

http.conf .cgiCGI

extension

- [SetHandler](#)



AddInputFilter

AddInputFilter <i>filter</i> [<i>;filter...</i>] <i>extension</i> [<i>extension</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
Apache 2.0.26

[AddInputFilter](#)*extension* [SetInputFilter](#) *extension*

filterextension *extension*

- [RemoveInputFilter](#)
- [SetInputFilter](#)



AddLanguage

AddLanguage <i>MIME-lang extension</i> [<i>extension</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

AddLanguage *extension* *MIME-lang* *extension*

```
AddEncoding x-compress .Z  
AddLanguage en .en  
AddLanguage fr .fr
```

xxxx.en.Z(xxxx.Z.en)

AddLanguage

```
AddLanguage en .en  
AddLanguage en-gb .en  
AddLanguage en-us .en
```

.enen-us

extension

- mod_negotiation



AddOutputFilter

AddOutputFilter <i>filter</i> [<i>;filter...</i>] <i>extension</i> [<i>extension</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
Apache 2.0.26

```
AddOutputFilter extension SetOutputFilter  
AddOutputFilterByType extension  
  
    .shtml    mod\_deflate
```

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

filterextension extension

- [RemoveOutputFilter](#)
- [SetOutputFilter](#)



AddType

AddType <i>MIME-type extension</i> [<i>extension</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

AddType *MIME-typeextension* *extension*(TypesConfig)

```
AddType image/gif .gif
```

```
AddType TypesConfig
```

extension

- DefaultType
- ForceType



DefaultLanguage

DefaultLanguage <i>MIME-lang</i>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

```
DefaultLanguageApache( <Directory>(  
AddLanguage.fr.de)  MIME-lang  
DefaultLanguage
```

[DefaultLanguageAddLanguage](#)

DefaultLanguage en

- [mod_negotiation](#)



path_info
ModMimeUsePathInfo On Off
ModMimeUsePathInfo Off
directory
(B)
mod_mime
Apache 2.0.41

```
ModMimeUsePathInfo mod_mime URL path_info Off
URL path_info
```

ModMimeUsePathInfo On

```
/bar/foo.shtml" /bar" ModMimeUsePathInfo On
mod_mime/bar/foo.shtml" AddOutputFilter INCLUDES
.shtml" INCLUDES ModMimeUsePathInfo INCLUDES
```

- [AcceptPathInfo](#)



MultiViews
MultiviewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters]
MultiviewsMatch NegotiatedOnly
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
Apache 2.0.26

MultiviewsMatch mod_negotiation Multiviews Multiviews
index.html index.

index.html.gz

NegotiatedOnly mod_mime

```
/    MultiviewsMatch Handlers Filters 500
index.html.cgi 1000 index.html.pl .cgi .asis
asis-handler .asis
```

mod_mime Any Apache 1.3.old.bak

Multiviews

MultiviewsMatch Handlers Filters

- [Options](#)
- [mod_negotiation](#)



RemoveCharset

RemoveCharset <i>extension</i> [<i>extension</i>] ...
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
Apache 2.0.24

RemoveCharset .htaccess

extension

RemoveCharset .html .shtml



RemoveEncoding

RemoveEncoding <i>extension</i> [<i>extension</i>] ...
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

RemoveEncoding .htaccess

/foo/.htaccess:

```
AddEncoding x-gzip .gz
AddType text/plain .asc
<Files *.gz.asc>
  RemoveEncoding .gz
</Files>
```

foo.gzgzip foo.gz.asc

[RemoveEncodingAddEncoding](#)
[AddEncoding](#)

[RemoveEncoding](#)

extension



RemoveHandler

<code>RemoveHandler <i>extension</i> [<i>extension</i>] ...</code>
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

`RemoveHandler .htaccess`

/foo/.htaccess

```
AddHandler server-parsed .html
```

/foo/bar/.htaccess

```
RemoveHandler .html
```

`/foo/bar.html` parsing([mod_include](#))

extension



RemoveInputFilter

<code>RemoveInputFilter <i>extension</i> [<i>extension</i>] ...</code>
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
Apache 2.0.26

`RemoveInputFilter .htaccess`

extension

- [AddInputFilter](#)
- [SetInputFilter](#)



RemoveLanguage

RemoveLanguage <i>extension</i> [<i>extension</i>] ...
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
Apache 2.0.24

RemoveLanguage .htaccess

extension



RemoveOutputFilter

<code>RemoveOutputFilter <i>extension</i> [<i>extension</i>] ...</code>
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime
2.0.26

`RemoveOutputFilter .htaccess`

extension

```
RemoveOutputFilter shtml
```

- [AddOutputFilter](#)



RemoveType

RemoveType <i>extension</i> [<i>extension</i>] ...
virtual host, directory, .htaccess
FileInfo
(B)
mod_mime

RemoveType .htaccess

```
/foo/.htaccess  
RemoveType .cgi
```

/foo/.cgi DefaultType

```
RemoveTypeAddType RemoveTypeAddType
```

extension



TypesConfig

mime.types
TypesConfig <i>file-path</i>
TypesConfig conf/mime.types
server config
(B)
mod_mime

TypesConfig MIME *File-path* ServerRoot mime.types
IANA <http://www.iana.org>
httpd.conf AddType mime.types

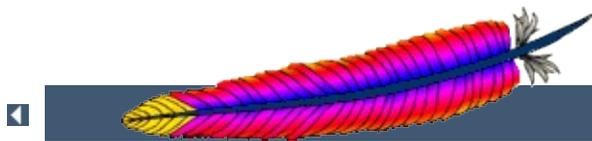
AddType

```
MIME-type [extension] ...
```

(#)

```
Apache HTTP mime.types(1) IANS(2) category/x-  
subtype
```

- mod_mime_magic



Apache HTTP Server 2.2

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

| | | 2006127 |

Apache mod_mime_magic

MIME
(E)
mime_magic_module
mod_mime_magic.c

Unixfile(1) [MIME](#)mod_mime""

Unixfile(1)"Magic""Magic"

[MimeMagicFile](#)



Magic4-5(#)

1	">">"																						
2	<table border="1"> <tr> <td>byte</td> <td></td> </tr> <tr> <td>short</td> <td>16</td> </tr> <tr> <td>long</td> <td>32</td> </tr> <tr> <td>string</td> <td></td> </tr> <tr> <td>date</td> <td>(UNIX/1970)</td> </tr> <tr> <td>beshort</td> <td>big-endian 16</td> </tr> <tr> <td>belong</td> <td>big-endian 32</td> </tr> <tr> <td>bedate</td> <td>big-endian 32</td> </tr> <tr> <td>leshort</td> <td>little-endian 16</td> </tr> <tr> <td>lelong</td> <td>little-endian 32</td> </tr> <tr> <td>ledate</td> <td>little-endian 32</td> </tr> </table>	byte		short	16	long	32	string		date	(UNIX/1970)	beshort	big-endian 16	belong	big-endian 32	bedate	big-endian 32	leshort	little-endian 16	lelong	little-endian 32	ledate	little-endian 32
byte																							
short	16																						
long	32																						
string																							
date	(UNIX/1970)																						
beshort	big-endian 16																						
belong	big-endian 32																						
bedate	big-endian 32																						
leshort	little-endian 16																						
lelong	little-endian 32																						
ledate	little-endian 32																						
3																							
4	MIME																						
5	MIME()																						

Magic

```
# 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
```

* .docMicrosoft WordFrame Maker()

```
# 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/mswoi
0 string \320\317\021\340\241\261 application/mswoi
0 string \333\245-\0\0\0 application/mswoi
```

MIMEgzip

```
# gzip (GNU zip, not to be confused with
# [Info-ZIP/PKWARE] zip archiver)

0 string \037\213 application/octet-stream x-gzip
```



web

file(1)webweb""



mod_mime_magic

mod_mime_magic: Magic NumberMIME
Copyright (c) 1996-1997 Cisco Systems, Inc.
Cisco19977ApacheCiscoApache
comp.sources.unixfile

- Copyright (c) Ian F. Darwin, 1987. Written by Ian F. Darwin.
(AT&T)

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

MrDarwin"file"

- Apache
- Apache
- ApacheApache()
- MagicApache APIrealloc()
- ()
- stdoutApacheMIME
-

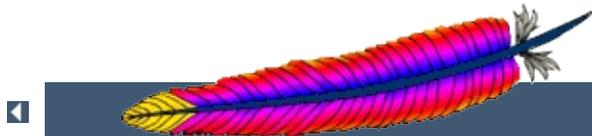


MimeMagicFile

MagicMIME
MimeMagicFile <i>file-path</i>
server config, virtual host
(E)
mod_mime_magic

```
MimeMagicFileMagic conf/magic ServerRoot
```

```
MimeMagicFile conf/magic
```



Apache HTTP Server 2.2

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

| | | 2006128 |

Apache mod_negotiation

(B)
negotiation_module
mod_negotiation.c

....

- (type-map)
- "MultiViews"([OptionsMultiViews](#))



RFC822(#)

Content-Encoding:

Apache [AddEncoding](#)compress x-compressgzipx-gzip
" x-"

Content-Language:

([RFC 1766](#)) en

Content-Length:

Content-Type:

[MIME](#)MIME" name=value"

level

text/html"2""0"

qs

0.01.0""jpegAsciijpeg qs

```
Content-Type: image/jpeg; qs=0.8
```

URI:

URIURL

Body:

2.0Body

```
Body:-----xyz-----  
<html>  
<body>  
<p>Content of the page.</p>  
</body>  
</html>
```

----xyz----



MULTIViews

```
MultiViewsOptionsMultiViews /some/dir/foo
/some/dir/foo foo.* foo.*
```

MultiViewsMatchApache



CacheNegotiatedDocs

CacheNegotiatedDocs On Off
CacheNegotiatedDocs Off
server config, virtual host
(B)
mod_negotiation
2.0

"On"

HTTP/1.0HTTP/1.1HTTP/1.1

2.0 CacheNegotiatedDocs



ForceLanguagePriority

ForceLanguagePriority None Prefer Fallback [Prefer Fallback]
ForceLanguagePriority Prefer
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_negotiation
Apache 2.0.30

ForceLanguagePriorityLanguagePriority

```
ForceLanguagePriority Prefer LanguagePriority  
HTTP"300") Accept-Languageen  
en
```

```
LanguagePriority en fr de  
ForceLanguagePriority Prefer
```

```
ForceLanguagePriority Fallback LanguagePriority  
HTTP"406") Accept-La  
LanguagePriority
```

```
LanguagePriority en fr de  
ForceLanguagePriority Fallback
```

```
PreferFallback LanguagePriority
```

- [AddLanguage](#)



LanguagePriority <i>MIME-lang</i> [<i>MIME-lang</i>] ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_negotiation

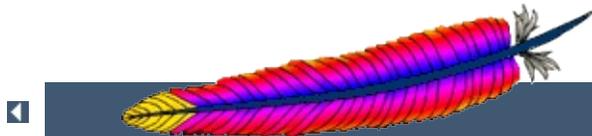
MultiViews LanguagePriority MIME-lang

LanguagePriority en fr de

foo.html foo.html.fr foo.html.de foo.html.fr

ForceLanguagePriorityNoneHTTP/1.1

- AddLanguage



Apache HTTP Server 2.2

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

| | | 2006128 |

Apache mod_nw_ssl

NetWareSSL
(B)
nwssl_module
mod_nw_ssl.c
NetWare

(port)SSLNetWareSSL



NWSSLTrustedCerts

NWSSLTrustedCerts <i>filename</i> [<i>filename</i>] ...
server config
(B)
mod_nw_ssl

(DER)SSL .der



NWSSLUpgradeable

SSL
NWSSLUpgradeable [IP-address:]portnumber
server config
(B)
mod_nw_ssl

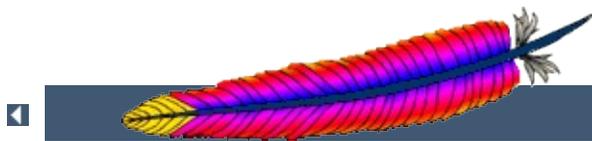
/SSL/ Listen



Configuration

SSL
SecureListen [<i>IP-address:</i>]portnumber Certificate-Name [MUTUAL]
server config
(B)
mod_nw_ssl

SSLeDirectorymutual



||||



Apache HTTP Server 2.2

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

|| |???

Apache mod_proxy

HTTP/1.1/
(E)
proxy_module
mod_proxy.c

ProxyRequests

Apache/ AJP13(Apache JServe Protocol v1.3), FTP,
CONNECT(SSL), HTTP/0.9, HTTP/1.0, HTTP/1.1

Apache(mod_proxy) mod_proxy http, mod_proxy ftp,
mod_proxy ajp, mod_proxy balancer,
mod_proxy connect mod_proxy(LoadModule)

mod_cache mod_sslSSLProxy*SSL/TLS



Apache(*forward*)(*reverse*)

(*origin server*)()

Internet(mod_cache)

ProxyRequests

(name-space)()

InternetURLwebwebURL

ProxyPass(RewriteRule[P]) ProxyRequests



mod_cache

```
ProxyRequests On  
ProxyVia On
```

```
<Proxy *>  
    Order deny,allow  
    Deny from all  
    Allow from internal.example.com  
</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
```



<Proxy>

```
<Proxy *>  
  Order Deny,Allow  
  Deny from all  
  Allow from 192.168.0  
</Proxy>
```

mod_authz_host

(ProxyRequests)(" ProxyPass)

ProxyRequests Off"



ProxyBlockIP



Apache([ProxyRemote](#))

[NoProxy](#)

WWW"http://somehost/"

http://somehost.example.com/

[Pro](#)



`mod_proxy(KeepAlive)HTTP/1.1 (KeepAlive)HTTP/1.0 SetEnv`

`force-proxy-request-1.0 proxy-nokeepalive`

```
<Location /buggyappserver/>
  ProxyPass http://buggyappserver:7001/foo/
  SetEnv force-proxy-request-1.0 1
  SetEnv proxy-nokeepalive 1
</Location>
```



(POST)HTTP(chunked transfer encoding) Content-Length
mod_proxy httpContent-Length proxy-
sendclContent-Length proxy-sendchunked



CONNECT
AllowCONNECT <i>port</i> [<i>port</i>] ...
AllowCONNECT 443 563
server config, virtual host
(E)
mod_proxy

AllowCONNECTCONNECT https http

https(443)snews(563) AllowCONNECT

mod_proxy_connect CONNECT



NoProxy

//
NoProxy <i>host</i> [<i>host</i>] ...
server config, virtual host
(E)
mod_proxy

Apache **NoProxyIP/** ProxyRemote

```
ProxyRemote * http://firewall.mycompany.com:81
NoProxy .mycompany.com 192.168.112.0/21
```

NoProxyhost

DNSDNS""

```
.com
.apache.org.
```

(DNSDNS"A"!)

```
DNS .MyDomain.com.mydomain.com.()DNS
```

bit()bit8bit

192.168192.168.0.0

" 192.168.0.0"16bit(255.255.0.0)

192.168.112.0/21

" 192.168.112.0/21"21bit(255.255.248.0)

32bit [IP](#)bit("0.0.0.0/0")" _Default_"IP

IP

IPDNS

192.168.123.7

IPDNSapache

DNSDNS [IP](#)() [IP](#)([IP](#))

prep.ai.mit.edu
www.apache.org

[IP](#)DNSPPPApache

DNS www.MyDomain.comwww.mydomain.com.()

- [DNS](#)



Proxy

<code><Proxy wildcard-url> ...</Proxy></code>
server config, virtual host
(E)
mod_proxy

`<Proxy>shell`

`yournetwork.example.com`

```
<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from yournetwork.example.com
</Proxy>
```

`example.comfooINCLUDES`

```
<Proxy http://example.com/foo/*>
  SetOutputFilter INCLUDES
</Proxy>
```



ProxyBadHeader

ProxyBadHeader IsError Ignore StartBody
ProxyBadHeader IsError
server config, virtual host
(E)
mod_proxy
Apache 2.0.44

ProxyBadHeader mod_proxy((:))

IsError

"502"(Bad Gateway)

Ignore

StartBody



ProxyBlock

<code>ProxyBlock * word host domain [word host domain]</code> ...
server config, virtual host
(E)
mod_proxy

ProxyBlock//HTTPHTTPSFTP IP

```
ProxyBlock joes-garage.com some-host.co.uk  
rocky.wotsamattau.edu
```

IP rocky.wotsamattau.edu
wotsamattauwotsamattau.edu

```
ProxyBlock *
```



ProxyDomain

ProxyDomain <i>Domain</i>
server config, virtual host
(E)
mod_proxy

Apache ProxyDomainapache *Domain*

```
ProxyRemote * http://firewall.mycompany.com:81
NoProxy .mycompany.com 192.168.112.0/21
ProxyDomain .mycompany.com
```



ProxyErrorOverride

ProxyErrorOverride On Off
ProxyErrorOverride Off
server config, virtual host
(E)
mod_proxy
Apache 2.0

([mod_includeSSI](#))(`"On"`SSI)



ProxyIOBufferSize

ProxyIOBufferSize <i>bytes</i>
ProxyIOBufferSize 8192
server config, virtual host
(E)
mod_proxy

ProxyIOBufferSize() 8192



ProxyMatch

<code><ProxyMatch regex> ...</ProxyMatch></code>
server config, virtual host
(E)
mod_proxy

`<ProxyMatch>`[<Proxy>](#)



ProxyMaxForwards

ProxyMaxForwards <i>number</i>
ProxyMaxForwards 10
server config, virtual host
(E)
mod_proxy
Apache 2.0

ProxyMaxForwardsDoS

ProxyMaxForwards 15



ProxyPass

URL
ProxyPass [<i>path</i>] ! <i>url</i> [<i>key=value key=value ...</i>]
server config, virtual host, directory
(E)
mod_proxy

URL *path* *url*/URL

```
ProxyPass    ProxyRequests    off
```

http://example.com/

```
ProxyPass /mirror/foo/ http://backend.example.com/
```

http://example.com/mirror/foo/bar
http://backend.example.com/bar

!"

```
ProxyPass /mirror/foo/i !  
ProxyPass /mirror/foo http://backend.example.com
```

/mirror/foo/ibackend.example.com/mirror/foo

```
ProxyPass
```

As of Apache 2.1, the ability to use pooled connections to a backend server is available. Using the `key=value` parameters it is possible to tune this connection pooling. The default for a `Hard` Maximum for the number of connections is the number of threads per process in the

active MPM. In the Prefork MPM, this is always 1, while with the Worker MPM it is controlled by the `ThreadsPerChild`.

Setting `min` will determine how many connections will always be open to the backend server. Upto the Soft Maximum or `smax` number of connections will be created on demand. Any connections above `smax` are subject to a time to live or `ttl`. Apache will never create more than the Hard Maximum or `max` connections to the backend server.

```
ProxyPass /example http://backend.example.com  
smax=5 max=20 ttl=120 retry=300
```

Parameter	Default	Description
<code>min</code>	0	Minumum number of connections that will always be open to the backend server.
<code>max</code>	1...n	Hard Maximum number of connections that will be allowed to the backend server. The default for a Hard Maximum for the number of connections is the number of threads per process in the active MPM. In the Prefork MPM, this is always 1, while with the Worker MPM it is controlled by the <code>ThreadsPerChild</code> . Apache will never create more than the Hard Maximum connections to the backend server.
<code>smax</code>	<code>max</code>	Upto the Soft Maximum number of connections will be created on demand. Any connections above <code>smax</code> are subject to a time to live or <code>ttl</code> .
<code>ttl</code>	-	Time To Live for the inactive connections above the <code>smax</code> connections in seconds. Apache will close all connections that has not been used inside that time period.

timeout	Timeout	Connection timeout in seconds. If not set the Apache will wait until the free connection is available. This directive is used for limiting the number of connections to the backend server together with max parameter.
acquire	-	If set this will be the maximum time to wait for a free connection in the connection pool. If there are no free connections in the pool the Apache will return SERVER_BUSY status to the client.
keepalive	Off	This parameter should be used when you have a firewall between your Apache and the backend server, who tend to drop inactive connections. This flag will tell the Operating System to send KEEP_ALIVE messages on inactive connections (interval depends on global OS settings, generally 120ms), and thus prevent the firewall to drop the connection. To enable keepalive set this property value to On.
retry	60	Connection pool worker retry timeout in seconds. If the connection pool worker to the backend server is in the error state, Apache will not forward any requests to that server until the timeout expires. This enables to shut down the backend server for maintenance, and bring it back online later.
loadfactor	1	Worker load factor. Used with BalancerMember. It is a number between 1 and 100 and defines the normalized weighted load applied to the worker.
route	-	Route of the worker when used inside load balancer. The route is a value appended to

		session id.
redirect	-	Redirection Route of the worker. This value is usually set dynamically to enable safe removal of the node from the cluster. If set all requests without session id will be redirected to the BalancerMember that has route parametar equal as this value.

If the Proxy directive scheme starts with the balancer :// then a virtual worker that does not really communicate with the backend server will be created. Instead it is responsible for the management of several "real" workers. In that case the special set of parameters can be add to this virtual worker.

Parameter	Default	Description
lbmethod	-	Balancer load-balance method. Select the load-balancing scheduler method to use. Either byrequests, to perform weighted request counting or bytraffic, to perform weighted traffic byte count balancing. Default is byrequests.
stickysession	-	Balancer sticky session name. The value is usually set to something like JSESSIONID PHPSESSIONID, and it depends on the backend application server that support sessions.
nofailover	Off	If set to On the session will break if the worker is in error state or disabled. Set this value to On if backend servers do not support session replication.
timeout	0	Balancer timeout in seconds. If set this will be the maximum time to wait for a free worker. Default is not to wait.

maxattempts	1	Maximum number of failover attempts before giving up.
-------------	---	---

```
ProxyPass /special-area
http://special.example.com/ smax=5 max=10
ProxyPass / balancer://mycluster
stickysession=jsessionid nofailover=On
<Proxy balancer://mycluster>
  BalancerMember http://1.2.3.4:8009
  BalancerMember http://1.2.3.5:8009 smax=10
  # Less powerful server, don't send as many
  requests there
  BalancerMember http://1.2.3.6:8009 smax=1
  loadfactor=20
</Proxy>
```

When used inside a [<Location>](#) section, the first argument is omitted and the local directory is obtained from the [<Location>](#).

If you require a more flexible reverse-proxy configuration, see the [RewriteRule](#) directive with the [P] flag.



ProxyPassReverse

HTTPURL
ProxyPassReverse <i>[path] url</i>
server config, virtual host, directory
(E)
mod_proxy

ApacheHTTPLocation, Content-Location, URIURLApache
HTTP

HTMLURLURLHTMLURLNick
[mod_proxy_html](#)

path *url*URL [ProxyPass](#)

`http://example.com/`

```
ProxyPass /mirror/foo/ http://backend.example.com/
ProxyPassReverse /mirror/foo/
http://backend.example.com/
ProxyPassReverseCookieDomain backend.example.com
public.example.com
ProxyPassReverseCookiePath / /mirror/foo/
```

```
http://example.com/mirror/foo/bar
http://backend.example.com/bar( ProxyPass)
backend.example.com http://backend.example.com/bar
http://backend.example.com/quuxApacheHTTP
http://example.com/mirror/foo/quux URL
```

[UseCanonicalName](#)

```
ProxyPassReversemod_rewrite(RewriteRule ... [P])
ProxyPass
```

<Location> <Location>



ProxyPassReverseCookieDomain

Adjusts the Domain string in Set-Cookie headers from a reverse-proxied server
ProxyPassReverseCookieDomain <i>internal-domain</i> <i>public-domain</i>
server config, virtual host, directory
(E)
mod_proxy

Usage is basically similar to [ProxyPassReverse](#), but instead of rewriting headers that are a URL, this rewrites the domain string in Set-Cookie headers.



ProxyPassReverseCookiePath

Adjusts the Path string in Set-Cookie headers from a reverse-proxied server
ProxyPassReverseCookiePath <i>internal-path public-path</i>
server config, virtual host, directory
(E)
mod_proxy

Usage is basically similar to [ProxyPassReverse](#), but instead of rewriting headers that are a URL, this rewrites the path string in Set-Cookie headers.



ProxyPreserveHost

HTTP
ProxyPreserveHost On Off
ProxyPreserveHost Off
server config, virtual host
(E)
mod_proxy
Apache 2.0.31

"Host:" **ProxyPass**

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.



ProxyReceiveBufferSize

HTTPFTP()
ProxyReceiveBufferSize <i>bytes</i>
ProxyReceiveBufferSize 0
server config, virtual host
(E)
mod_proxy

ProxyReceiveBufferSizeHTTPFTP(TCP/IP) 512 " 0"

ProxyReceiveBufferSize 2048



ProxyRemote

ProxyRemote <i>match remote-server</i>
server config, virtual host
(E)
mod_proxy

*matchURLURL" *" remote-serverURL*

remote-server = scheme://hostname[:port]

scheme http

```
ProxyRemote http://goodguys.com/  
http://mirrorguys.com:8000  
ProxyRemote * http://cleversite.com  
ProxyRemote ftp http://ftpproxy.mydomain.com:8080
```

HTTPFTP

webURL



ProxyRemoteMatch

ProxyRemoteMatch <i>regex remote-server</i>
server config, virtual host
(E)
mod_proxy

ProxyRemoteMatch [ProxyRemoteURL](#)



ProxyRequests

()
ProxyRequests On Off
ProxyRequests Off
server config, virtual host
(E)
mod_proxy

Apache(OffProxyPass)

Off

HTTPFTP mod_proxy httpmod_proxy ftp

ProxyRequests



ProxyTimeout

ProxyTimeout <i>seconds</i>
ProxyTimeout 300
server config, virtual host
(E)
mod_proxy
Apache 2.0.31

/

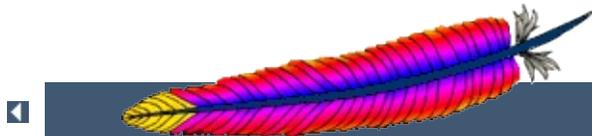


ProxyVia

Via
ProxyVia On Off Full Block
ProxyVia Off
server config, virtual host
(E)
mod_proxy

" Via:" [RFC 2616](#)(HTTP/1.1)14.45" Via:"

- Off " Via:"
- On" Via:"
- Full " Via:"Apache" Via:"
- Block " Via:" Via:"



||||



Apache HTTP Server 2.2

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

|| < > | ??? |

Apache mod_proxy_ajp

mod_proxy Apache JServ Protocol
(E)
proxy_ajp_module
proxy_ajp.c
Apache 2.1

This module *requires* the service of [mod_proxy](#). It provides support for the Apache JServ Protocol version 1.3 (hereafter *AJP13*).

Thus, in order to get the ability of handling AJP13 protocol, [mod_proxy](#) and [mod_proxy_ajp](#) have to be present in the server.

Internet



Structure of the Protocol

AJP13 protocol is packet-oriented. A binary format was presumably chosen over the more readable plain text for reasons of performance. The web server communicates with the servlet container over TCP connections. To cut down on the expensive process of socket creation, the web server will attempt to maintain persistent TCP connections to the servlet container, and to reuse a connection for multiple request/response cycles.

Once a connection is assigned to a particular request, it will not be used for any others until the request-handling cycle has terminated. In other words, requests are not multiplexed over connections. This makes for much simpler code at either end of the connection, although it does cause more connections to be open at once.

Once the web server has opened a connection to the servlet container, the connection can be in one of the following states:

- Idle
No request is being handled over this connection.
- Assigned
The connecton is handling a specific request.

Once a connection is assigned to handle a particular request, the basic request informaton (e.g. HTTP headers, etc) is sent over the connection in a highly condensed form (e.g. common strings are encoded as integers). Details of that format are below in Request Packet Structure. If there is a body to the request (`content-length > 0`), that is sent in a separate packet immediately after.

At this point, the servlet container is presumably ready to start processing the request. As it does so, it can send the following messages back to the web server:

- SEND_HEADERS

- Send a set of headers back to the browser.
- SEND_BODY_CHUNK
Send a chunk of body data back to the browser.
 - GET_BODY_CHUNK
Get further data from the request if it hasn't all been transferred yet. This is necessary because the packets have a fixed maximum size and arbitrary amounts of data can be included the body of a request (for uploaded files, for example). (Note: this is unrelated to HTTP chunked transfer).
 - END_RESPONSE
Finish the request-handling cycle.

Each message is accompanied by a differently formatted packet of data. See Response Packet Structures below for details.



There is a bit of an XDR heritage to this protocol, but it differs in lots of ways (no 4 byte alignment, for example).

Byte order: I am not clear about the endian-ness of the individual bytes. I'm guessing the bytes are little-endian, because that's what XDR specifies, and I'm guessing that sys/socket library is magically making that so (on the C side). If anyone with a better knowledge of socket calls can step in, that would be great.

There are four data types in the protocol: bytes, booleans, integers and strings.

Byte

A single byte.

Boolean

A single byte, `1 = true`, `0 = false`. Using other non-zero values as true (i.e. C-style) may work in some places, but it won't in others.

Integer

A number in the range of `0` to `2^16` (32768). Stored in 2 bytes with the high-order byte first.

String

A variable-sized string (length bounded by `2^16`). Encoded with the length packed into two bytes first, followed by the string (including the terminating `\0`). Note that the encoded length does **not** include the trailing `\0` -- it is like `strlen`. This is a touch confusing on the Java side, which is littered with odd autoincrement statements to skip over these terminators. I believe the reason this was done was to allow the C code to be extra efficient when reading strings which the servlet container is sending back -- with the terminating `\0` character, the C code can pass around references into a single buffer, without copying. If the `\0` was missing, the C code would have to copy things out in

order to get its notion of a string.

Packet Size

According to much of the code, the max packet size is $8 * 1024$ bytes (8K). The actual length of the packet is encoded in the header.

Packet Headers

Packets sent from the server to the container begin with 0x1234. Packets sent from the container to the server begin with AB (that's the ASCII code for A followed by the ASCII code for B). After those first two bytes, there is an integer (encoded as above) with the length of the payload. Although this might suggest that the maximum payload could be as large as 2^{16} , in fact, the code sets the maximum to be 8K.

<i>Packet Format (Server->Container)</i>					
Byte	0	1	2	3	4...(n+3)
Contents	0x12	0x34	Data Length (n)		Data

<i>Packet Format (Container->Server)</i>					
Byte	0	1	2	3	4...(n+3)
Contents	A	B	Data Length (n)		Data

For most packets, the first byte of the payload encodes the type of message. The exception is for request body packets sent from the server to the container -- they are sent with a standard packet header (0x1234 and then length of the packet), but without any prefix code after that.

The web server can send the following messages to the servlet container:

Code	Type of Packet	Meaning
2	Forward Request	Begin the request-processing cycle with the following data
7	Shutdown	The web server asks the container to shut itself down.
8	Ping	The web server asks the container to take control (secure login phase).
10	CPing	The web server asks the container to respond quickly with a CPong.
none	Data	Size (2 bytes) and corresponding body data.

To ensure some basic security, the container will only actually do the Shut down if the request comes from the same machine on which it's hosted.

The first Data packet is send immediatly after the Forward Request by the web server.

The servlet container can send the following types of messages to the webserver:

Code	Type of Packet	Meaning
3	Send Body Chunk	Send a chunk of the body from the servlet container to the web server (and presumably, onto the browser).
4	Send Headers	Send the response headers from the servlet container to the web server (and presumably, onto the browser).
5	End Response	Marks the end of the response (and thus the request-handling cycle).
6	Get Body	Get further data from the request if it hasn't all

	Chunk	been transferred yet.
9	CPong Reply	The reply to a CPing request

Each of the above messages has a different internal structure, detailed below.



Request Header Structure

For messages from the server to the container of type *Forward Request*:

```
AJP13_FORWARD_REQUEST :=
    prefix_code      (byte) 0x02 = JK_AJP13_FORWARD_REQUEST
    method           (byte)
    protocol         (string)
    req_uri          (string)
    remote_addr      (string)
    remote_host      (string)
    server_name      (string)
    server_port      (integer)
    is_ssl           (boolean)
    num_headers      (integer)
    request_headers *(req_header_name req_header_value)
    attributes       *(attribute_name attribute_value)
    request_terminator (byte) 0xFF
```

`request_headers` have the following structure:

```
req_header_name :=
    sc_req_header_name | (string) [see below for how
sc_req_header_name := 0xA0xx (integer)
req_header_value := (string)
```

`attributes` are optional and have the following structure:

```
attribute_name := sc_a_name | (sc_a_req_attribute string)
attribute_value := (string)
```

Not that the all-important header is content-length, because it determines whether or not the container looks for another packet immediately.

Detailed description of the elements of Forward Request

Request prefix

For all requests, this will be 2. See above for details on other Prefix codes.

Method

The HTTP method, encoded as a single byte:

Command Name	Code
OPTIONS	1
GET	2
HEAD	3
POST	4
PUT	5
DELETE	6
TRACE	7
PROPFIND	8
PROPPATCH	9
MKCOL	10
COPY	11
MOVE	12
LOCK	13
UNLOCK	14
ACL	15

REPORT	16
VERSION-CONTROL	17
CHECKIN	18
CHECKOUT	19
UNCHECKOUT	20
SEARCH	21
MKWORKSPACE	22
UPDATE	23
LABEL	24
MERGE	25
BASELINE_CONTROL	26
MKACTIVITY	27

Later version of ajp13, will transport additional methods, even if they are not in this list.

protocol, req_uri, remote_addr, remote_host, server_name, server_port, is_ssl

These are all fairly self-explanatory. Each of these is required, and will be sent for every request.

Headers

The structure of request_headers is the following: First, the number of headers num_headers is encoded. Then, a series of header name req_header_name / value req_header_value pairs follows. Common header names are encoded as integers, to save space. If the header name is not in the list of basic headers, it is encoded normally (as a string, with prefixed length). The list of common headers sc_req_header_name and their codes is as follows (all are case-sensitive):

--	--	--

Name	Code value	Code name
accept	0xA001	SC_REQ_ACCEPT
accept-charset	0xA002	SC_REQ_ACCEPT_CHARSET
accept-encoding	0xA003	SC_REQ_ACCEPT_ENCODING
accept-language	0xA004	SC_REQ_ACCEPT_LANGUAGE
authorization	0xA005	SC_REQ_AUTHORIZATION
connection	0xA006	SC_REQ_CONNECTION
content-type	0xA007	SC_REQ_CONTENT_TYPE
content-length	0xA008	SC_REQ_CONTENT_LENGTH
cookie	0xA009	SC_REQ_COOKIE
cookie2	0xA00A	SC_REQ_COOKIE2
host	0xA00B	SC_REQ_HOST
pragma	0xA00C	SC_REQ_PRAGMA
referer	0xA00D	SC_REQ_REFERER
user-agent	0xA00E	SC_REQ_USER_AGENT

The Java code that reads this grabs the first two-byte integer and if it sees an '0xA0' in the most significant byte, it uses the integer in the second byte as an index into an array of header names. If the first byte is not 0xA0, it assumes that the two-byte integer is the length of a string, which is then read in.

This works on the assumption that no header names will have length greater than 0x9999 (==0xA000 - 1), which is perfectly reasonable, though somewhat arbitrary.

The content-length header is extremely important. If it is present and non-zero, the container assumes that the request has a body (a POST request, for example), and immediately reads a separate packet off the input stream to get that body.

Attributes

The attributes prefixed with a ? (e.g. ?context) are all optional. For each, there is a single byte code to indicate the type of attribute, and then a string to give its value. They can be sent in any order (though the C code always sends them in the order listed below). A special terminating code is sent to signal the end of the list of optional attributes. The list of byte codes is:

Information	Code Value	Note
?context	0x01	Not currently implemented
?servlet_path	0x02	Not currently implemented
?remote_user	0x03	
?auth_type	0x04	
?query_string	0x05	
?jvm_route	0x06	
?ssl_cert	0x07	
?ssl_cipher	0x08	
?ssl_session	0x09	
?req_attribute	0x0A	Name (the name of the attribute follows)
?ssl_key_size	0x0B	
are_done	0xFF	request_terminator

contextservlet_path are not currently set by the C code, and most of the Java code completely ignores whatever is sent over for those fields (and some of it will actually break if a string is sent along after one of those codes). I don't know if this is a bug or an unimplemented feature or just vestigial code, but it's missing from both sides of the connection.

remote_userauth_type presumably refer to HTTP-level authentication, and communicate the remote user's username and the type of authentication used to establish their identity (e.g. Basic,

Digest).

`query_string`, `ssl_cert`, `ssl_cipher`, and `ssl_session` refer to the corresponding pieces of HTTP and HTTPS.

`jvm_route`, is used to support sticky sessions -- associating a user's session with a particular Tomcat instance in the presence of multiple, load-balancing servers.

Beyond this list of basic attributes, any number of other attributes can be sent via the `req_attribute` code `0x0A`. A pair of strings to represent the attribute name and value are sent immediately after each instance of that code. Environment values are passed in via this method.

Finally, after all the attributes have been sent, the attribute terminator, `0xFF`, is sent. This signals both the end of the list of attributes and also then end of the Request Packet.



for messages which the container can send back to the server.

```
AJP13_SEND_BODY_CHUNK :=
    prefix_code      3
    chunk_length    (integer)
    chunk           *(byte)

AJP13_SEND_HEADERS :=
    prefix_code      4
    http_status_code (integer)
    http_status_msg  (string)
    num_headers      (integer)
    response_headers *(res_header_name header_value)

res_header_name :=
    sc_res_header_name | (string) [see below for how

sc_res_header_name := 0xA0 (byte)

header_value := (string)

AJP13_END_RESPONSE :=
    prefix_code      5
    reuse            (boolean)

AJP13_GET_BODY_CHUNK :=
    prefix_code      6
    requested_length (integer)
```

Details:

Send Body Chunk

The chunk is basically binary data, and is sent directly back to the browser.

Send Headers

The status code and message are the usual HTTP things (e.g. 200 OK). The response header names are encoded the same way the request header names are. See `header_encoding` above for details about how the codes are distinguished from the strings.

The codes for common headers are:

Name	Code value
Content-Type	0xA001
Content-Language	0xA002
Content-Length	0xA003
Date	0xA004
Last-Modified	0xA005
Location	0xA006
Set-Cookie	0xA007
Set-Cookie2	0xA008
Servlet-Engine	0xA009
Status	0xA00A
WWW-Authenticate	0xA00B

After the code or the string header name, the header value is immediately encoded.

End Response

Signals the end of this request-handling cycle. If the `reuse` flag is `true` (`==1`), this TCP connection can now be used to handle new incoming requests. If `reuse` is `false` (anything other than 1 in the actual C code), the connection should be closed.

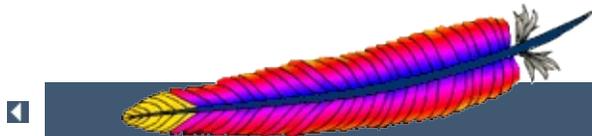
Get Body Chunk

The container asks for more data from the request (If the body was too large to fit in the first packet sent over or when the request is chunked). The server will send a body packet back with an amount of data which is the minimum of the `request_length`, the maximum send body size (8186 (8 Kbytes - 6)), and the number of bytes actually left to send from the request body.

If there is no more data in the body (i.e. the servlet container is trying to read past the end of the body), the server will send back an *empty* packet, which is a body packet with a payload length of 0.

(0x12, 0x34, 0x00, 0x00)

||||



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_proxy_balancer

mod_proxy
(E)
proxy_balancer_module
proxy_balancer.c
Apache 2.1

This module *requires* the service of [mod_proxy](#). It provides load balancing support for HTTP, FTPAJP13 protocols

Thus, in order to get the ability of load balancing, [mod_proxy](#) [mod_proxy_balancer](#) have to be present in the server.

Internet



At present, there are 2 load balancer scheduler algorithms available for use: Request Counting and Weighted Traffic Counting. These are controlled via the `lbmethod` value of the Balancer definition. See the [Proxy](#) directive for more information.



Request Scheduling Algorithm

Enabled via `lbmethod=byrequests`, the idea behind this scheduler is that we distribute the requests among the various workers to ensure that each gets their configured share of the number of requests. It works as follows:

lbfactor is how much we expect this worker to work, or the worker's work quota. This is a normalized value representing their "share" of the amount of work to be done.

lbstatus is how urgent this worker has to work to fulfill its quota of work.

worker is a member of the load balancer, usually a remote host serving one of the supported protocols.

We distribute each worker's work quota to the worker, and then look which of them needs to work most urgently (biggest *lbstatus*). This worker is then selected for work, and its *lbstatus* reduced by the total work quota we distributed to all workers. Thus the sum of all *lbstatus* does not change(*) and we distribute the requests as desired.

If some workers are disabled, the others will still be scheduled correctly.

```
for each worker in workers
  worker lbstatus += worker lbfactor
  total factor    += worker lbfactor
  if worker lbstatus > candidate lbstatus
    candidate = worker

candidate lbstatus -= total factor
```

If a balancer is configured as follows:

worker	a	b	c	d
--------	---	---	---	---

lbfactor	25	25	25	25
lbstatus	0	0	0	0

And *b* gets disabled, the following schedule is produced:

worker	a	b	c	d
lbstatus	-50	0	25	25
lbstatus	-25	0	-25	50
lbstatus	0	0	0	0
(repeat)				

That is it schedules: *a c d a c d a c d ...* Please note that:

worker	a	b	c	d
lbfactor	25	25	25	25

Has the exact same behavior as:

worker	a	b	c	d
lbfactor	1	1	1	1

This is because all values of *lbfactor* are normalized with respect to the others. For:

worker	a	b	c
lbfactor	1	4	1

worker *b* will, on average, get 4 times the requests that *ac* will.

The following asymmetric configuration works as one would expect:

worker	a	b
lbfactor	70	30

lbstatus	-30	30
lbstatus	40	-40
lbstatus	10	-10
lbstatus	-20	20
lbstatus	-50	50
lbstatus	20	-20
lbstatus	-10	10
lbstatus	-40	40
lbstatus	30	-30
lbstatus	0	0
(repeat)		

That is after 10 schedules, the schedule repeats and 7 *a* are selected with 3 *b* interspersed.



Weighted Round Counting Algorithm

Enabled via `lbmethod=bytraffic`, the idea behind this scheduler is very similar to the Request Counting method, with the following changes:

lbfactor is how much traffic, in bytes, we want this worker to handle. This is also a normalized value representing their "share" of the amount of work to be done, but instead of simply counting the number of requests, we take into account the amount of traffic this worker has seen.

If a balancer is configured as follows:

worker	a	b	c
lbfactor	1	2	1

Then we mean that we want *b* to process twice the amount of bytes than *a* or *c* should. It does not necessarily mean that *b* would handle twice as many requests, but it would process twice the I/O. Thus, the size of the request and response are applied to the weighting and selection algorithm.



This module *requires* the service of `mod_status`. Balancer manager enables dynamic update of balancer members. You can use balancer manager to change the balance factor or a particular member, or put it in the off line mode.

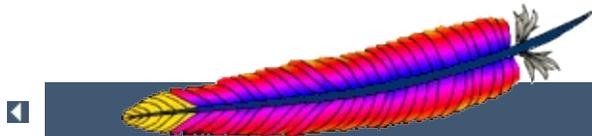
Thus, in order to get the ability of load balancer management, `mod_status``mod_proxy_balancer` have to be present in the server.

To enable load balancer management for browsers from the foo.com domain add this code to your `httpd.conf` configuration file

```
<Location /balancer-manager>
  SetHandler balancer-manager

  Order Deny,Allow
  Deny from all
  Allow from .foo.com
</Location>
```

You can now access load balancer manager by using a Web browser to access the page `http://your.server.name/balancer-manager`



Apache HTTP Server 2.2

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

| | | 2006128 |

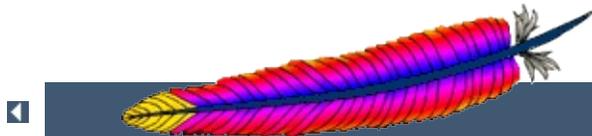
Apache mod_proxy_connect

mod_proxy HTTP CONNECT
(E)
proxy_connect_module
proxy_connect.c

[mod_proxy](#)HTTP CONNECTSSL

CONNECT [mod_proxy](#)[mod_proxy_connect](#)

Internet



Apache HTTP Server 2.2

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

| | | 2006128 |

Apache mod_proxy_ftp

mod_proxyFTP
(E)
proxy_ftp_module
proxy_ftp.c

FTP [mod_proxyFTP](#)

[mod_proxy](#) [mod_proxy_ftp](#)

FTPGET

Internet



`mimeapplication/octet-stream`

`application/octet-stream bin dms lha lzh exe class t`

`DefaultType application/octet-stream`



FTP ASCII(binary)" ;type=a" mod_proxyASCII
FTPASCII



mod_proxyFTPGETFTPApacheHTTP(POSTPUT)



FTP URIhome"/../"(.)FTPApache FTP" *Squid*
%2f hack" [Squid Proxy Cache](#) /%2f"FTP" /"(home)
/etc/motd URL

ftp://user@host/%2f/etc/motd



FTP Apache URL Apache FTP

```
user: anonymous  
password: apache_proxy@
```

FTP

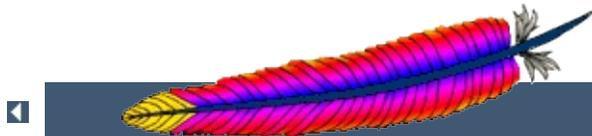
URL

```
ftp://username@host/myfile
```

FTP() Apache" 401"()/

```
ftp://username:password@host/myfile
```

Apachebase64ApacheFTPHTTPFTP(FTP)



Apache HTTP Server 2.2

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

| | | 2006128 |

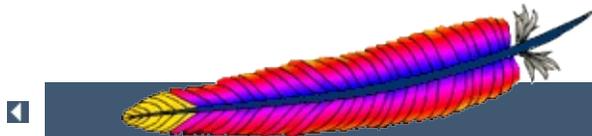
Apache mod_proxy_http

mod_proxyHTTP
(E)
proxy_http_module
proxy_http.c

[mod_proxyHTTP](#) HTTP/1.0, HTTP/1.1 [mod_proxy_http](#) HTTP/0.9,
[mod_cache](#)

HTTP [mod_proxymod_proxy_http](#)

Internet



||||

◀ Apache HTTP Server 2.2

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

|| < > | ??? |

Apache mod_rewrite

URL
(E)
rewrite_module
mod_rewrite.c
Apache 1.3

URLURLURLHTTPURL

URL()(httpd.conf)(.htaccess)

[URL](#)



Apache 1.3.20 *TestStringSubstitution(\)*

Substitution

" \ \$"mod_rewrite



()CGI/SSI SCRIPT_URLSCRIPT_URI
SCRIPT_NAMESCRIPT_FILENAME

URI/URL URI/URLURL

```
SCRIPT_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.l  
SCRIPT_FILENAME=/u/rse/.www/index.html  
SCRIPT_URL=/u/rse/  
SCRIPT_URI=http://en1.engelschall.com/u/rse/
```



[URLURLURL](#)



URL
RewriteBase <i>URL-path</i>
directory, .htaccess
FileInfo
(E)
mod_rewrite

```
RewriteBaseURL    RewriteRule(.htaccess)
" RewriteBase physical-directory-path"
```

```
URLURLURLURL          URL!
RewriteBaseURL
```

```
URL RewriteBase .htaccessRewriteRule
```

```
#
# /abc/def/.htaccess -- per-dir config file for directory
# Remember: /abc/def is the physical path of /xyz, i.e.
#           has a 'Alias /xyz /abc/def' directive
#

RewriteEngine On

# let the server know that we were reached via /xyz :
# via the physical path prefix /abc/def
RewriteBase    /xyz

# now the rewriting rules
RewriteRule    ^oldstuff\.html$ newstuff.html
```

/xyz/oldstuff.html/abc/def/newstuff.html

For Apache Hackers

Request:

/xyz/oldstuff.html

Internal Processing:

```
/xyz/oldstuff.html      -> /abc/def/oldstuff.html (f
/abc/def/oldstuff.html -> /abc/def/newstuff.html (f
/abc/def/newstuff.html -> /xyz/newstuff.html (f
/xyz/newstuff.html     -> /abc/def/newstuff.html (f
```

Result:

/abc/def/newstuff.html

()ApacheApacheApacheApache



RewriteCond <i>TestString CondPattern</i>
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_rewrite

RewriteCond [RewriteRule](#) RewriteCond URIpattern

TestString

- RewriteRule

\$N

(0 <= N <= 9) (RewriteRule) RewriteCondpattern(!)

- RewriteCond

%N

(1 <= N <= 9) RewriteCond(!)

- RewriteMap

`\${mapname: key|default}`

[RewriteMap](#)

-

%{ *NAME_OF_VARIABLE* }

NAME_OF_VARIABLE

HTTP headers:	connection & request:	
HTTP_USER_AGENT	REMOTE_ADDR	

HTTP_REFERER HTTP_COOKIE HTTP_FORWARDED HTTP_HOST HTTP_PROXY_CONNECTION HTTP_ACCEPT	REMOTE_HOST REMOTE_PORT REMOTE_USER REMOTE_IDENT REQUEST_METHOD SCRIPT_FILENAME PATH_INFO QUERY_STRING AUTH_TYPE	
server internals:	date and time:	specials:
DOCUMENT_ROOT SERVER_ADMIN SERVER_NAME SERVER_ADDR SERVER_PORT SERVER_PROTOCOL SERVER_SOFTWARE	TIME_YEAR TIME_MON TIME_DAY TIME_HOUR TIME_MIN TIME_SEC TIME_WDAY TIME	API_VERSION THE_REQUEST REQUEST_URI REQUEST_FILENAME IS_SUBREQ HTTPS

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:

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 [mod_ssl](#) is loaded).

Special Notes:

1. The variables `SCRIPT_FILENAME` and `REQUEST_FILENAME` contain the same value, *i.e.*, the value of the `filename` field of the internal `request_rec` structure of the Apache server. The first name is just the commonly known CGI variable name while the second is the consistent counterpart to `REQUEST_URI` (which contains the value of the `uri` field of `request_rec`).
2. There is the special format: `%{ENV:variable}` where *variable* can be any environment variable. This is looked-up via internal Apache structures and (if not found there) via `getenv()` from the Apache server process.

3. There is the special format: `%{SSL:variable}` where *variable* is the name of an [SSL environment variable](#); this 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. There is the special format: `%{HTTP:header}` where *header* can be any HTTP MIME-header name. This is looked-up from the HTTP request. Example: `%{HTTP:Proxy-Connection}` is the value of the HTTP header "Proxy-Connection".
5. There is the special format `%{LA-U:variable}` for look-aheads which perform an internal (URL-based) sub-request to determine the final value of *variable*. Use this when you want to use a variable for rewriting which is actually set later in an API phase and thus is not available at the current stage. For instance when you want to rewrite according to the `REMOTE_USER` variable from within the per-server context (`httpd.conf` file) you have to use `%{LA-U:REMOTE_USER}` because this variable is set by the authorization phases which come *after* the URL translation phase where `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}` there.
6. There is the special format: `%{LA-F:variable}` which performs 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, *i.e.*, a regular expression which is applied to the current instance of the *TestString*, *i.e.*, *TestString* is evaluated and then 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**' (is lexically lower)
Treats the *CondPattern* as a plain string and compares it lexically to *TestString*. True if *TestString* is lexically lower than *CondPattern*.
 - '**>CondPattern**' (is lexically greater)
Treats the *CondPattern* as a plain string and compares it lexically to *TestString*. True if *TestString* is lexically greater than *CondPattern*.
 - '**=CondPattern**' (is lexically equal)
Treats the *CondPattern* as a plain string and compares it lexically to *TestString*. True if *TestString* is lexically equal to *CondPattern*, i.e the two strings are exactly equal (character by character). If *CondPattern* is just "" (two quotation marks) this compares *TestString* to the empty string.
 - '**-d**' (is **directory**)
Treats the *TestString* as a pathname and tests if it exists and is a directory.
 - '**-f**' (is **regular file**)
Treats the *TestString* as a pathname and tests if it exists and is a regular file.
 - '**-s**' (is **regular file with size**)
Treats the *TestString* as a pathname and tests if it exists and is a regular file with size greater than zero.
 - '**-l**' (is **symbolic link**)
Treats the *TestString* as a pathname and tests if it exists and

is a symbolic link.

- **'-x'** (has executable permissions)
Treats the *TestString* as a pathname and tests if it exists and has execution permissions. These permissions are determined depending on the underlying OS.
- **'-F'** (is existing file via subrequest)
Checks if *TestString* is a valid file and accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to determine the check, so use it with care because it decreases your servers performance!
- **'-U'** (is existing URL via subrequest)
Checks if *TestString* is a valid URL and accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to determine the check, so use it with care because it decreases your server's performance!

Notice

All of these tests can also be prefixed by an exclamation mark (!) to negate their meaning.

Additionally you can set special flags for *CondPattern* by appending

[flags]

as the third argument to the `RewriteCond` directive. *Flags* is a comma-separated list of the following flags:

- **'nocase|NC'** (no case)
This makes the test case-insensitive, *i.e.*, there is no difference between 'A-Z' and 'a-z' both in the expanded *TestString* and the *CondPattern*. This flag is effective only for comparisons between

TestStringCondPattern. It has no effect on filesystem and subrequest checks.

- **'ornext |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
```

Without this flag you would have to write the cond/rule 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

RewriteCond %{HTTP_USER_AGENT} ^Lynx.*
RewriteRule ^/$ /homepage.min.html

RewriteRule ^/$ /homepage.std.html
```

Interpretation: If you use Netscape Navigator as your browser (which identifies itself as 'Mozilla'), then you get the max homepage, which includes Frames, *etc.* If you use the Lynx browser (which is Terminal-based), then you get the min homepage, which contains no images, no tables, *etc.* If you use any other browser you get the standard homepage.



RewriteEngine

Enables or disables runtime rewriting engine
RewriteEngine on off
RewriteEngine off
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_rewrite

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.



REWRITELOCK

Sets the name of the lock file used for RewriteMap synchronization
<code>RewriteLock <i>file-path</i></code>
server config
(E)
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.



Sets the name of the file used for logging rewrite engine processing
<code>RewriteLog file-path</code>
server config, virtual host
(E)
mod_rewrite

`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!`

See the [Apache 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.

```
RewriteLog
"/usr/local/var/apache/logs/rewrite.log"
```



RewriteLogLevel

Sets the verbosity of the log file used by the rewrite engine
<code>RewriteLogLevel <i>Level</i></code>
<code>RewriteLogLevel 0</code>
server config, virtual host
(E)
mod_rewrite

`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!

```
RewriteLogLevel 3
```



RewriteMap

Defines a mapping function for key-lookup
<code>RewriteMap MapName MapType:MapSource</code>
server config, virtual host
(E)
mod_rewrite
The choice of different dbm types is available in Apache 2.0.41

`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.

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 *MapTypeMapSource* 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

```
##
##  map.txt -- rewriting map
##

Ralf.S.Engelschall    rse    # Bastard Operator Fro
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";
tie (%DB, 'NDBM_File', $dbmmap, O_RDWR|O_TRUNC|O_CREAT, 0666)
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 exists:

- **toupper:**
Converts the looked up key to all upper case.
- **tolower:**

Converts the looked up key to all lower case.

- **escape:**
Translates special characters in the looked up key to hex-encodings.
- **unescape:**
Translates hex-encodings in the looked up 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 the language of your choice, but the result has to be a executable (*i.e.*, either object-code or a script with the magic cookie trick '#!/path/to/interpreter' as the first line).

This program is started once at startup of the Apache servers and then communicates with the rewriting engine over its `stdin` `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:

```
#!/usr/bin/perl
$| = 1;
while (<STDIN>) {
    # ...put here any transformations or lookups..
    print $_;
}
```

But be very careful:

1. "*Keep it simple, stupid*" (KISS), because if this program hangs it will hang the Apache server when the rule occurs.
2. Avoid one common mistake: never do buffered I/O on `stdout`! This will cause a deadlock! Hence the "`$|=1`" in the above example...
3. Use the `RewriteLock` directive to define a lockfile `mod_rewrite` can use to synchronize the communication to the program. By default no such synchronization takes place.

`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.

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!



RewriteOptions

Sets some special options for the rewrite engine
RewriteOptions <i>Options</i>
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_rewrite
MaxRedirects is no longer available in version 2.1

RewriteOptions directive sets some special options for the current per-server or per-directory configuration. The *Option* string can be currently only one:

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.



Defines rules for the rewriting engine
RewriteRule <i>Pattern Substitution</i>
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_rewrite
The cookie-flag is available in Apache 2.0.40

RewriteRule directive is the real rewriting workhorse. The directive can occur more than once. Each directive then defines one single rewriting rule. The **definition order** of these rules is **important**, because this order is used when applying the rules at run-time.

Pattern is a perl compatible regular expression which gets applied to the current URL. Here "current" means the value of the URL when this rule gets applied. This may not be the originally requested URL, because any number of rules may already have matched and made alterations to it.

Some hints about the syntax of [regular expressions](#):

Text:

```
.           Any single character
[chars]    Character class: One of chars
[^chars]   Character class: None of chars
text1|text2 Alternative: text1 or text2
```

Quantifiers:

```
?          0 or 1 of the preceding text
*          0 or N of the preceding text (N > 0)
+          1 or N of the preceding text (N > 1)
```

Grouping:`(text)`

Grouping of text

(either to set the borders of an alternative or for making backreferences where the `Nth` can be used on the RHS of a RewriteRule with `%N`)

Anchors:`^`

Start of line anchor

`$`

End of line anchor

Escaping:`\char`

escape that particular char

(for instance to specify the chars `].[|`)

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.*) have a look at the following dedicated book on this topic:

Mastering Regular Expressions, 2nd Edition

Jeffrey E.F. Friedl

O'Reilly & Associates, Inc. 2002

ISBN 0-596-00289-0

Additionally in `mod_rewrite` the NOT character (`!`) is a possible pattern prefix. This gives you the ability 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.

Notice

When using the NOT character to negate a pattern you cannot

have grouped wildcard parts in the pattern. This is impossible because when the pattern does NOT match, there are no contents for the groups. In consequence, if negated patterns are used, you cannot use \$N in the substitution string!

Substitution of a rewriting rule is the string which is substituted for (or replaces) the original URL for which *Pattern* matched. Beside plain text you can use

1. back-references \$N to the RewriteRule pattern
2. back-references %N to the last matched RewriteCond pattern
3. server-variables as in rule condition test-strings (%{VARNAME})
4. [mapping-function](#) calls (\${mapname:key|default})

Back-references are \$N (N=0..9) identifiers which will be replaced by the contents of the Nth group of the matched *Pattern*. The server-variables are the same as for the *TestString* of a RewriteCond directive. The mapping-functions come from the RewriteMap directive and are explained there. These three types of variables are expanded in the order of the above list.

As already mentioned above, all the rewriting rules are applied to the *Substitution* (in the order of definition in the config file). The URL is **completely replaced** by the *Substitution* and the rewriting process goes on until there are no more rules unless explicitly terminated by a **L** flag - see below.

There is a special substitution string named '-' which means: **NO substitution!** Sounds silly? No, it is useful to provide rewriting rules which **only** match some URLs but do no substitution, in conjunction with the **C** (chain) flag to be able to have more than one pattern to be applied before a substitution occurs.

Query String

Pattern will not match 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. Just use a question mark inside the substitution string to indicate that the following stuff should be re-injected into the query string. When you want to erase an existing query string, end the substitution string with just the question mark. To combine a new query string with an old one, use the `[QSA]` flag (see below).

Substitution of Absolute URLs

There is a special feature: When you prefix a substitution field with `http://thishost[:thisport]` then **mod_rewrite** automatically strips it out. This auto-reduction on implicit external redirect URLs is a useful and important feature when used in combination with a mapping-function which generates the hostname part. Have a look at the first example in the example section below to understand this.

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 (see below).

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 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, *etc.*). This has the following effect: if a rule matches, then processing continues as

usual, *i.e.*, the flag has no effect. If the rule does **not** match, then all following chained rules are skipped. For instance, use it to remove the ".www" part inside a per-directory rule set when you let an external redirect happen (where the ".www" part should not to occur!).

- **'cookie|CO=NAME:VAL:domain[:lifetime[:path]]'** (set **cookie**)
This sets a cookie on 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 $\$N\%N$ which will be expanded. You can use this flag more than once to set more than one variable. The variables can be later dereferenced in many situations, but usually from within XSSI (via `<! --#echo var="VAR" -->`) or CGI (`$ENV{'VAR'}`). Additionally you can dereference it in a following RewriteCond pattern via `%{ENV:VAR}`. Use this to strip but remember information from URLs.
- **'forbidden|F'** (force URL to be **forbidden**)
This forces the current URL to be forbidden, *i.e.*, 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, *i.e.*, it immediately sends back a HTTP response of 410 (GONE). Use this flag to mark pages which no longer exist as gone.
- **'handler|H=Content-handler'** (force **Content handler**)
Force the Content-handler of the target file to be *Content-handler*. For instance, this can be used to simulate the `mod_alias` directive `ScriptAlias` which internally forces all

files inside the mapped directory to have a handler of "cgi-script".

- **'last | L'** (last rule)

Stop the rewriting process here and don't apply any more rewriting rules. This corresponds to the Perl `last` command or the `break` command from the C language. 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/www/`'.

- **'next | N'** (next round)

Re-run the rewriting process (starting again with the first rewriting rule). Here the URL to match is again not the original URL but the URL from the last rewriting rule. This corresponds to the Perl `next` command or the `continue` command from the C language. Use this flag to restart the rewriting process, *i.e.*, to immediately go to the top of the loop.

But be careful not to create an infinite loop!

- **'nocase | NC'** (no case)

This makes the *Pattern* case-insensitive, *i.e.*, there is no difference between 'A-Z' and 'a-z' when *Pattern* is matched against the current URL.

- **'noescape | NE'** (no URI escaping of output)

This flag keeps `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 being done. 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'** (used only if **no** internal **sub**-request)
This flag forces the rewriting engine to skip a rewriting 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 even sometimes causes a failure to if the complete set of rules are applied. Use this flag to exclude some rules.
Use the following rule for your decision: whenever you prefix some URLs with CGI-scripts to force them to be processed by the CGI-script, the chance is high that you will run into problems (or even overhead) on sub-requests. In these cases, use this flag.
- **'proxy|P'** (force **proxy**)
This flag forces the substitution part to be internally forced as a proxy request and immediately (*i.e.*, rewriting rule processing stops here) put through the [proxy module](#). You have to 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 get an error from the proxy module. Use this flag to achieve a more powerful implementation of the [ProxyPass](#) directive, to map some remote stuff into the namespace of the local server.
`mod_proxy` must be enabled in order to use this flag.
- **'passthrough|PT'** (**pass** through to next handler)
This flag forces the rewriting 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 be able to post-process the output of `RewriteRule` directives by `Alias`, `ScriptAlias`, `Redirect`, *etc.* directives from other URI-to-filename translators. A trivial example to show the semantics: If you want to rewrite `/abc` to `/def` via the rewriting engine of `mod_rewrite` and then

/def to /ghi with mod_alias:

```
RewriteRule ^/abc(.*) /def$1 [PT]
Alias /def /ghi
```

If you omit the PT flag then mod_rewrite will do its job fine, *i.e.*, it rewrites uri=/abc/... to filename=/def/... as a full API-compliant URI-to-filename translator should do. Then mod_alias comes and tries to do a URI-to-filename transition which will not work.

Note: **You have to use this flag if you want to intermix directives of different modules which contain URL-to-filename translators.** The typical example is the use of mod_aliasmod_rewrite..

- **'qsappend|QSA'** (query string append)
This flag forces the rewriting engine to append a query string part in the substitution string to the existing one 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 a external redirection. If no *code* is given a HTTP response of 302 (MOVED TEMPORARILY) is used. If you want to use other response codes in the range 300-400 just specify them as a number or use one of the following symbolic names: temp (default), permanent, seeother. Use it for rules which should canonicalize the URL and give it back to the client, translate "/~" into "/u/" or always append a slash to /u/user, etc.
Note: When you use this flag, make sure that the substitution field is a valid URL! If not, you are redirecting to an invalid location! And remember that this flag itself only prefixes the URL with http://thishost[:thisport]/, rewriting continues.

Usually you also want to stop and do the redirection immediately. To stop the rewriting you also have to provide the 'L' flag.

- '**skip|S=num**' (skip next rule(s))

This flag forces the rewriting engine to skip the next *num* rules in sequence when 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*. For instance, this can be used to setup 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]
```

Never forget that *Pattern* is applied to a complete URL in per-server configuration files. **But 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, because without this prefix stripping you 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!

To enable the rewriting engine for per-directory configuration files you need to set "RewriteEngine On" in these files "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.

Here are all possible substitution combinations and their meanings:

**Inside per-server configuration (httpd.conf)
for request "GET /somepath/pathinfo":**

Given Rule	Result
----- ^/somepath(.*) otherpath\$1	not su
^/somepath(.*) otherpath\$1 [R]	not su
^/somepath(.*) otherpath\$1 [P]	not su
----- ^/somepath(.*) /otherpath\$1	/other
^/somepath(.*) /otherpath\$1 [R]	http:// via ex
^/somepath(.*) /otherpath\$1 [P]	not su
----- ^/somepath(.*) http://thishost/otherpath\$1	/other
^/somepath(.*) http://thishost/otherpath\$1 [R]	http:// via ex
^/somepath(.*) http://thishost/otherpath\$1 [P]	not su
----- ^/somepath(.*) http://otherhost/otherpath\$1	http://

		via ex
^/somepath(.*)	http://otherhost/otherpath\$1 [R]	http:// via ex (the
^/somepath(.*)	http://otherhost/otherpath\$1 [P]	http:// via in

**Inside per-directory configuration for /somepath
(i.e., file .htaccess in dir /physical/path/to/somepath
containing RewriteBase /somepath)
for request "GET /somepath/localpath/pathinfo":**

Given Rule	Result
----- ^localpath(.*) otherpath\$1	----- /somep
^localpath(.*) otherpath\$1 [R]	http:// via ex
^localpath(.*) otherpath\$1 [P]	not su
----- ^localpath(.*) /otherpath\$1	----- /other
^localpath(.*) /otherpath\$1 [R]	http:// via ex
^localpath(.*) /otherpath\$1 [P]	not su
----- ^localpath(.*) http://thishost/otherpath\$1	----- /other
^localpath(.*) http://thishost/otherpath\$1 [R]	http:// via ex

```

^localpath(.*) http://thishost/otherpath$1 [P] not su
-----
^localpath(.*) http://otherhost/otherpath$1      http://
via ex

^localpath(.*) http://otherhost/otherpath$1 [R] http://
via ex
(the |

^localpath(.*) http://otherhost/otherpath$1 [P] http://
via it

```

Example:

We want to rewrite URLs of the form

/ Language /~ Realname /.../ File

into

/u/ Username /.../ File . Language

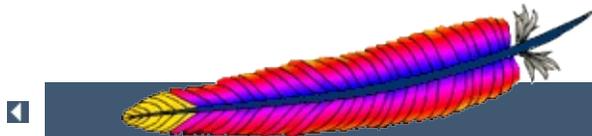
We take the rewrite mapfile from above and save it under `/path/to/file/map.txt`. Then we only have to add the following lines to the Apache server configuration file:

```

RewriteLog    /path/to/file/rewrite.log
RewriteMap    real-to-user                txt:/path/to/f
RewriteRule   ^/([\^/]+)/~([\^/]+)/(.*)$ /u/${real-to-u

```

||||



◀ Apache HTTP Server 2.2

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

| | | 2006129 |

Apache mod_setenvif

(B)
setenvif_module
mod_setenvif.c

mod_setenvif

mozillaMSIE

netscape

```
BrowserMatch ^Mozilla netscape  
BrowserMatch MSIE !netscape
```



BrowserMatch

User-Agent
BrowserMatch <i>regex</i> <i>[!]env-variable[=value]</i> <i>[!]env-variable[=value]</i> ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_setenvif

BrowserMatchSetEnvIf User-Agent

```
BrowserMatchNoCase Robot is_a_robot  
SetEnvIfNoCase User-Agent Robot is_a_robot
```

```
BrowserMatch ^Mozilla forms jpeg=yes  
browser=netscape  
BrowserMatch "^Mozilla/[2-3]" tables agif frames  
javascript  
BrowserMatch MSIE !javascript
```



BrowserMatchNoCase

User-Agent
BrowserMatchNoCase <i>regex</i> <i>[!]env-variable[=value]</i> <i>[!]env-variable[=value]</i> ...
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_setenvif

BrowserMatchNoCaseBrowserMatch

```
BrowserMatchNoCase mac platform=macintosh
BrowserMatchNoCase win platform=windows
```

BrowserMatchBrowserMatchNoCaseSetEnvIf SetEnvIfNoCase

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```



<code>SetEnvIf attribute regex [!]env-variable[=value] [!]env-variable[=value]] ...</code>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_setenvif

SetEnvIf attribute

1. HTTP([RFC2616](#)) Host, User-Agent, Referer, Accept-Language
2.
 - Remote_Host ()
 - Remote_Addr IP
 - Server_Addr IP(2.0.43)
 - Request_Method (GET, POST)
 - Request_Protocol ("HTTP/0.9", "HTTP/1.0", "HTTP/1.1")
 - Request_URI HTTP(URL)
3. `SetEnvIf [NoCase]"attribute"`

regexPerlregexattribute

1. `varname`
2. `!varname`
3. `varname=value`

`varname"1" varname() varnamevalue2.0.51`
Apache `value$1..$9 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(Referer`
`www.mydomain.com)`

`HAVE_TS("TS"[a-z])`

- [Apache](#)

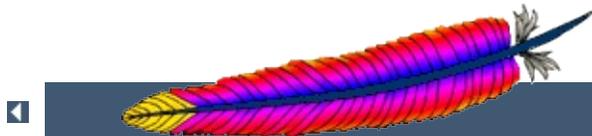


<code>SetEnvIfNoCase <i>attribute</i> <i>regex</i> [!]env-variable[=<i>value</i>] [[!]env-variable[=<i>value</i>]] ...</code>
server config, virtual host, directory, .htaccess
FileInfo
(B)
mod_setenvif

SetEnvIfNoCase SetEnvIf

```
SetEnvIfNoCase Host Apache\.Org site=apache
```

```
site" apache "(" Host:"" Apache.Org"" apache.org"
)
```



Apache HTTP Server 2.2

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

| | | 2006129 |

Apache mod_so

DSO
(E)
so_module
mod_so.c
Windows()

Apache [DSO](#)

Unix(.so)Windows .so.dll

Apache1.3Apache2.0



Apache1.3.15Windowsmod_foo.so

ApacheAPIUnixWindowsUnixWindows

UnixWindowsApacheUnix ConfigureApacheCore
(symbols) os\win32\modules.c

(DLL) **LoadModule**DLLApache

DLL(module record)DLL()

AP_MODULE_DECLARE_DATA(Apache)(module record)

```
module foo_module;
```

```
module AP_MODULE_DECLARE_DATA foo_module;
```

WindowsUnix .DEF

DLLlibhttpd.dlllibhttpd.libApache"modules".dsp
.dsp

DLL modules **LoadModule**



LoadFile

<code>LoadFile filename [filename] ...</code>
server config
(E)
mod_so

Filename[ServerRoot](#)

```
LoadFile libexec/libxmlparse.so
```



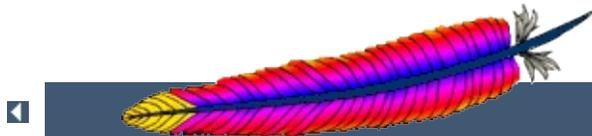
LoadModule

LoadModule <i>module filename</i>
server config
(E)
mod_so

filename *module* *module* ([Module Identifier](#))

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

[ServerRoot](#)



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_speling

URL
(E)
speling_module
mod_speling.c

Requests to documents sometimes cannot be served by the core apache server because the request was misspelled or miscapitalized. This module addresses this problem by trying to find a matching document, even after all other modules gave up. It does its work by comparing each document name in the requested directory against the requested document name **without regard to case**, and allowing **up to one misspelling** (character insertion / omission / transposition or wrong character). A list is built with all document names which were matched using this strategy.

If, after scanning the directory,

- no matching document was found, Apache will proceed as usual and return a "document not found" error.
- only one document is found that "almost" matches the request, then it is returned in the form of a redirection response.
- more than one document with a close match was found, then the list of the matches is returned to the client, and the client can select the correct candidate.



Enables the spelling module
CheckSpelling on off
CheckSpelling Off
server config, virtual host, directory, .htaccess
Options
(E)
mod_speling
CheckSpelling was available as a separately available module for Apache 1.1, but was limited to miscapitalizations. As of Apache 1.3, it is part of the Apache distribution. Prior to Apache 1.3.2, the CheckSpelling directive was only available in the "server" and "virtual host" contexts.

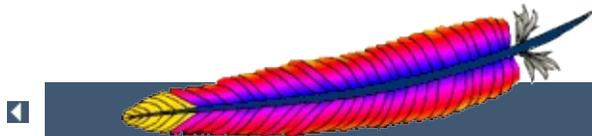
This directive enables or disables the spelling module. When enabled, keep in mind that

- the directory scan which is necessary for the spelling correction will have an impact on the server's performance when many spelling corrections have to be performed at the same time.
- the document trees should not contain sensitive files which could be matched inadvertently by a spelling "correction".
- the module is unable to correct misspelled user names (as in `http://my.host/~apahce/`), just file names or directory names.
- spelling corrections apply strictly to existing files, so a request for the `<Location /status>` may get incorrectly treated as the negotiated file `"/stats.html"`.

`mod_speling` should not be enabled in [DAV](#) enabled directories, because it will try to "spell fix" newly created resource names against existing filenames, e.g., when trying to upload a new document `doc43.html` it might redirect to an existing document `doc34.html`,

which is not what was intended.





| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_ssl

(SSL)(TLS)
(E)
ssl_module
mod_ssl.c

This module provides SSL v2/v3 and TLS v1 support for the Apache HTTP Server. It was contributed by Ralf S. Engeschall based on his mod_ssl project and originally derived from work by Ben Laurie.

This module relies on [OpenSSL](#) to provide the cryptography engine.

Further details, discussion, and examples are provided in the [SSL documentation](#).



This module provides a lot of SSL information as additional environment variables to the SSI and CGI namespace. The generated variables are listed in the table below. For backward compatibility the information can be made available under different names, too. Look in the [Compatibility](#) chapter for details on the compatibility variables.

Variable Name:	Value Type:	Description:
HTTPS	flag	HTTPS is being used.
SSL_PROTOCOL	string	The SSL protocol version (SSLv2, SSLv3, TLSv1)
SSL_SESSION_ID	string	The hex-encoded SSL session id
SSL_CIPHER	string	The cipher specification name
SSL_CIPHER_EXPORT	string	true if cipher is an export cipher
SSL_CIPHER_USEKEYSIZE	number	Number of cipher bits (actually used)
SSL_CIPHER_ALGKEYSIZE	number	Number of cipher bits (possible)
SSL_COMPRESS_METHOD	string	SSL compression method negotiated
SSL_VERSION_INTERFACE	string	The mod_ssl program version
SSL_VERSION_LIBRARY	string	The OpenSSL program version
SSL_CLIENT_M_VERSION	string	The version of the client certificate
SSL_CLIENT_M_SERIAL	string	The serial of the client certificate

SSL_CLIENT_S_DN	string	Subject DN in client's certificate
SSL_CLIENT_S_DN_x509	string	Component of client's Subject DN
SSL_CLIENT_I_DN	string	Issuer DN of client's certificate
SSL_CLIENT_I_DN_x509	string	Component of client's Issuer DN
SSL_CLIENT_V_START	string	Validity of client's certificate (start time)
SSL_CLIENT_V_END	string	Validity of client's certificate (end time)
SSL_CLIENT_V_REMAIN	string	Number of days until client's certificate expires
SSL_CLIENT_A_SIG	string	Algorithm used for the signature of client's certificate
SSL_CLIENT_A_KEY	string	Algorithm used for the public key of client's certificate
SSL_CLIENT_CERT	string	PEM-encoded client certificate
SSL_CLIENT_CERT_CHAIN_ <i>n</i>	string	PEM-encoded certificates in client certificate chain
SSL_CLIENT_VERIFY	string	NONE, SUCCESS, GENEROUSFAILED : <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

x509 specifies a component of an X.509 DN; one of C, ST, L, O, OU, CN, T, I, G, S, D, UID, Email. In Apache 2.1 and later, x509 may also include a numeric _n suffix. If the DN in question contains multiple attributes of the same name, this suffix is used as an index to select a particular attribute. For example, where the server certificate subject DN included two OU fields, SSL_SERVER_S_DN_OU_0SSL_SERVER_S_DN_OU_1 could be used to reference each.

SSL_CLIENT_V_REMAIN is only available in version 2.1 and later.



Custom Log Format

When `mod_ssl` is built into Apache or at least loaded (under DSO situation) additional functions exist for the [Custom Log Format](#) of `mod_log_config`. First there is an additional `"%{varname}x"` extension format function which can be used to expand any variables provided by any module, especially those provided by `mod_ssl` which can you find in the above table.

For backward compatibility there is additionally a special `"%{name}c"` cryptography format function provided. Information about this function is provided in the [Compatibility](#) chapter.

```
CustomLog logs/ssl_request_log \ "%t %h %  
{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```



File of concatenated PEM-encoded CA Certificates for Client Auth
<code>SSLCACertificateFile</code> <i>file-path</i>
server config, virtual host
(E)
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](#).

```
SSLCACertificateFile  
/usr/local/apache2/conf/ssl.crt/ca-bundle-  
client.crt
```



Directory of PEM-encoded CA Certificates for Client Auth
SSLCACertificatePath <i>directory-path</i>
server config, virtual host
(E)
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. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

```
SSLCACertificatePath
/usr/local/apache2/conf/ssl.crt/
```



File of concatenated PEM-encoded CA Certificates for defining acceptable CA names
<code>SSLCADNRequestFile</code> <i>file-path</i>
server config, virtual host
(E)
mod_ssl

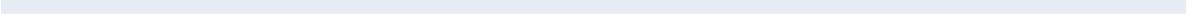
When a client certificate is requested by mod_ssl, a list of *acceptable Certificate Authority names* is sent to the client in the SSL handshake. These CA names can be used by the client to select an appropriate client certificate out of those it has available.

If neither of the directives [SSLCADNRequestPath](#) [SSLCADNRequestFile](#) are given, then the set of acceptable CA names sent to the client is the names of all the CA certificates given by the [SSLCACertificateFileSSLCACertificatePath](#) directives; in other words, the names of the CAs which will actually be used to verify the client certificate.

In some circumstances, it is useful to be able to send a set of acceptable CA names which differs from the actual CAs used to verify the client certificate - for example, if the client certificates are signed by intermediate CAs. In such cases, [SSLCADNRequestPath](#) and/or [SSLCADNRequestFile](#) can be used; the acceptable CA names are then taken from the complete set of certificates in the directory and/or file specified by this pair of directives.

[SSLCADNRequestFile](#) must specify an *all-in-one* file containing a concatenation of PEM-encoded CA certificates.

```
SSLCADNRequestFile /usr/local/apache2/conf/ca-
names.crt
```



Directory of PEM-encoded CA Certificates for defining acceptable CA names
SSLCADNRequestPath <i>directory-path</i>
server config, virtual host
(E)
mod_ssl

This optional directive can be used to specify the set of *acceptable CA names* which will be sent to the client when a client certificate is requested. See the [SSLCADNRequestFile](#) directive for more details.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value.N*. And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

```
SSLCADNRequestPath /usr/local/apache2/conf/ca-names.crt/
```



File of concatenated PEM-encoded CA CRLs for Client Auth
<code>SSLCARevocationFile</code> <i>file-path</i>
server config, virtual host
(E)
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](#).

```
SSLCARevocationFile
/usr/local/apache2/conf/ssl.crl/ca-bundle-
client.crl
```



Directory of PEM-encoded CA CRLs for Client Auth
SSLCARevocationPath <i>directory-path</i>
server config, virtual host
(E)
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. Use the `Makefile` which comes with [mod_ssl](#) to accomplish this task.

```
SSLCARevocationPath  
/usr/local/apache2/conf/ssl.crl/
```



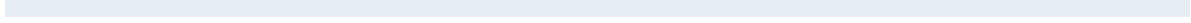
File of PEM-encoded Server CA Certificates
<code>SSLCertificateChainFile</code> <i>file-path</i>
server config, virtual host
(E)
mod_ssl

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 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.

```
SSLCertificateChainFile  
/usr/local/apache2/conf/ssl.crt/ca.crt
```



Server PEM-encoded X.509 Certificate file
<code>SSLCertificateFile</code> <i>file-path</i>
server config, virtual host
(E)
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.

```
SSLCertificateFile  
/usr/local/apache2/conf/ssl.crt/server.crt
```



Server PEM-encoded Private Key file
<code>SSLCertificateKeyFile</code> <i>file-path</i>
server config, virtual host
(E)
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.

```
SSLCertificateKeyFile  
/usr/local/apache2/conf/ssl.key/server.key
```



Cipher Suite available for negotiation in SSL handshake
SSLCipherSuite <i>cipher-spec</i>
SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
server config, virtual host, directory, .htaccess
AuthConfig
(E)
mod_ssl

This complex directive uses a colon-separated *cipher-spec* string consisting of OpenSSL cipher specifications to configure the Cipher Suite the client is permitted to negotiate in the SSL handshake phase. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured Cipher Suite after the HTTP request was read but before the HTTP response is sent.

An SSL cipher specification in *cipher-spec* is composed of 4 major attributes plus a few extra minor ones:

- *Key Exchange Algorithm:*
RSA or Diffie-Hellman variants.
- *Authentication Algorithm:*
RSA, Diffie-Hellman, DSS or none.
- *Cipher/Encryption Algorithm:*
DES, Triple-DES, RC4, RC2, IDEA or none.
- *MAC Digest Algorithm:*
MD5, SHA or SHA1.

An SSL cipher can also be an export cipher and is either a SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3). To specify which ciphers to use, one can either specify all the Ciphers, one at a

time, or use aliases to specify the preference and order for the ciphers (see [Table 1](#)).

Tag	Description
<i>Key Exchange Algorithm:</i>	
kRSA	RSA key exchange
kDhr	Diffie-Hellman key exchange with RSA key
kDHd	Diffie-Hellman key exchange with DSA key
kEDH	Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)
<i>Authentication Algorithm:</i>	
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication
<i>Cipher Encoding Algorithm:</i>	
eNULL	No encoding
DES	DES encoding
3DES	Triple-DES encoding
RC4	RC4 encoding
RC2	RC2 encoding
IDEA	IDEA encoding
<i>MAC Digest Algorithm:</i>	
MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	SHA hash function
<i>Aliases:</i>	
SSLv2	all SSL version 2.0 ciphers
SSLv3	all SSL version 3.0 ciphers
TLSv1	

	all TLS version 1.0 ciphers
EXP	all export ciphers
EXPORT40	all 40-bit export ciphers only
EXPORT56	all 56-bit export ciphers only
LOW	all low strength ciphers (no export, single DES)
MEDIUM	all ciphers with 128 bit encryption
HIGH	all ciphers using Triple-DES
RSA	all ciphers using RSA key exchange
DH	all ciphers using Diffie-Hellman key exchange
EDH	all ciphers using Ephemeral Diffie-Hellman key exchange
ADH	all ciphers using Anonymous Diffie-Hellman key exchange
DSS	all ciphers using DSS authentication
NULL	all ciphers using no encryption

Now where this becomes interesting is that these can be put together to specify the order and ciphers you wish to use. To speed this up there are also aliases (SSLv2, SSLv3, TLSv1, EXP, LOW, MEDIUM, HIGH) for certain groups of ciphers. These tags can be joined together with prefixes to form the *cipher-spec*. Available prefixes are:

- none: add cipher to list
- +: add ciphers to list and pull them to 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:-
NULL-SHA          SSLv3 Kx=RSA      Au=RSA  Enc=
NULL-MD5          SSLv3 Kx=RSA      Au=RSA  Enc=
EDH-RSA-DES-CBC3-SHA  SSLv3 Kx=DH       Au=RSA  Enc=
...
EXP-RC4-MD5       SSLv3 Kx=RSA(512) Au=RSA  Enc=
EXP-RC2-CBC-MD5   SSLv2 Kx=RSA(512) Au=RSA  Enc=
EXP-RC4-MD5       SSLv2 Kx=RSA(512) Au=RSA  Enc=
```

The complete list of particular RSA & DH ciphers for SSL is given in [Table 2](#).

```
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-	SSLv2	RSA	RSA	RC2(128)	MD5	

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 -	SSLv3	DH	RSA	DES(56)	SHA1	

DES - CBC - SHA						
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



Enable use of a cryptographic hardware accelerator
SSLCryptoDevice <i>engine</i>
SSLCryptoDevice builtin
server config
(E)
mod_ssl
Available if mod_ssl is built using -DSSL_ENGINE_EXPERIMENTAL

This directive enables use of a cryptographic hardware accelerator board to offload some of the SSL processing overhead. This directive can only be used if the SSL toolkit is built with "engine" support; OpenSSL 0.9.7 and later releases have "engine" support by default, the separate "-engine" releases of OpenSSL 0.9.6 must be used.

To discover which engine names are supported, run the command "openssl engine".

```
# For a Broadcom accelerator:
SSLCryptoDevice ubsec
```



SSLEngine

SSL Engine Operation Switch
SSLEngine on off optional
SSLEngine off
server config, virtual host
(E)
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.

```
<VirtualHost _default_:443>  
SSLEngine on  
...  
</VirtualHost>
```

In Apache 2.1 and later, `SSLEngine` can be set to `optional`. This enables support for [RFC 2817](#), Upgrading to TLS Within HTTP/1.1. At this time no web browsers support RFC 2817.



SSLHonorCipherOrder

Option to prefer the server's cipher preference order
<code>SSLHonorCipherOrder</code> <i>flag</i>
server config, virtual host
(E)
mod_ssl
Apache 2.1 and later, if using OpenSSL 0.9.7 or later

When choosing a cipher during an SSLv3 or TLSv1 handshake, normally the client's preference is used. If this directive is enabled, the server's preference will be used instead.

`SSLHonorCipherOrder on`



Semaphore for internal mutual exclusion of operations
SSLMutex <i>type</i>
SSLMutex none
server config
(E)
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" SysV 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 `fcntlflock`, 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](#).

```
SSLMutex file:/usr/local/apache/logs/ssl_mutex
```



Options

Configure various SSL engine run-time options
<code>SSLOptions [+ -]option ...</code>
server config, virtual host, directory, .htaccess
Options
(E)
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`, `SSL_CLIENT_CERT_CHAIN_n` (with $n = 0,1,2,..$). These contain the PEM-encoded X.509 Certificates of server and client for the current HTTPS connection and can be used by CGI scripts for deeper Certificate checking. Additionally all other certificates of the client certificate chain are provided, too. This bloats up the environment a little bit which is why you have to use this option to enable it on demand.

- `FakeBasicAuth`

When this option is enabled, the Subject Distinguished Name (DN) of the Client X509 Certificate is translated into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is just the Subject of the Client's X509 Certificate (can be determined by running OpenSSL's `openssl x509` command: `openssl x509 -noout -subject -in certificate.crt`). Note that no password is obtained from the user. Every entry in the user file needs this password: `"xXj31ZMTZzkVA"`, which is the DES-encrypted version of the word "password". Those who live under MD5-based encryption (for instance under FreeBSD or BSD/OS, etc.) should use the following MD5 hash of the same word: `"$1$0XLyS...$0wx8s2/m9/gfkcRVXzgoE/"`.

- `StrictRequire`

This *forces* forbidden access when `SSLRequireSSL` `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` `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.

```
SSLOptions +FakeBasicAuth -StrictRequire
<Files ~ "\.(cgi|html)$">
SSLOptions +StdEnvVars +CompatEnvVars -
ExportCertData
<Files>
```



Type of pass phrase dialog for encrypted private keys
SSLPassPhraseDialog <i>type</i>
SSLPassPhraseDialog builtin
server config
(E)
mod_ssl

When Apache starts up it has to read the various Certificate (see [SSLCertificateFile](#)) and Private Key (see [SSLCertificateKeyFile](#)) files of the SSL-enabled virtual servers. Because for security reasons the Private Key files are usually encrypted, mod_ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. This query can be done in two ways which can be configured by *type*:

- **builtin**

This is the default where an interactive terminal dialog occurs at startup time just before Apache detaches from the terminal. Here the administrator has to manually enter the Pass Phrase for each encrypted Private Key file. Because a lot of SSL-enabled virtual hosts can be configured, the following reuse-scheme is used to minimize the dialog: When a Private Key file is encrypted, all known Pass Phrases (at the beginning there are none, of course) are tried. If one of those known Pass Phrases succeeds no dialog pops up for this particular Private Key file. If none succeeded, another Pass Phrase is queried on the terminal and remembered for the next round (where it perhaps can be reused).

This scheme allows mod_ssl to be maximally flexible (because for N encrypted Private Key files you *can* use N different Pass Phrases - but then you have to enter all of them, of course) while minimizing the terminal dialog (i.e. when you use a single Pass Phrase for all N Private Key files this Pass Phrase is queried only

once).

- `|/path/to/program [args...]`
This mode allows an external program to be used which acts as a pipe to a particular input device; the program is sent the standard prompt text used for the `builtin` mode on `stdin`, and is expected to write password strings on `stdout`. If several passwords are needed (or an incorrect password is entered), additional prompt text will be written subsequent to the first password being returned, and more passwords must then be written back.
- `exec:/path/to/program`
Here an external program is configured which is called at startup for each encrypted Private Key file. It is called with two arguments (the first is of the form "servername:portnumber", the second is either "RSA" 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.



SSLPassPhraseDialog
exec:/usr/local/apache/sbin/pp-filter



Configure usable SSL protocol flavors
SSLProtocol [+ -] <i>protocol</i> ...
SSLProtocol all
server config, virtual host
Options
(E)
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.

```
# enable SSLv3 and TLSv1, but not SSLv2  
SSLProtocol all -SSLv2
```



SSLProxyCACertificateFile

File of concatenated PEM-encoded CA Certificates for Remote Server Auth
SSLProxyCACertificateFile <i>file-path</i>
server config, virtual host
(E)
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](#).

```
SSLProxyCACertificateFile
/usr/local/apache2/conf/ssl.crt/ca-bundle-remote-
server.crt
```



Directory of PEM-encoded CA Certificates for Remote Server Auth
SSLProxyCACertificatePath <i>directory-path</i>
server config, virtual host
(E)
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.

```
SSLProxyCACertificatePath
/usr/local/apache2/conf/ssl.crt/
```



SSLProxyCARevocationFile

File of concatenated PEM-encoded CA CRLs for Remote Server Auth
<code>SSLProxyCARevocationFile <i>file-path</i></code>
server config, virtual host
(E)
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](#).

```
SSLProxyCARevocationFile
/usr/local/apache2/conf/ssl.crl/ca-bundle-remote-
server.crl
```



Directory of PEM-encoded CA CRLs for Remote Server Auth
SSLProxyCAREvocationPath <i>directory-path</i>
server config, virtual host
(E)
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.

```
SSLProxyCAREvocationPath  
/usr/local/apache2/conf/ssl.crl/
```



SSLProxyCipherSuite

Cipher Suite available for negotiation in SSL proxy handshake
SSLProxyCipherSuite <i>cipher-spec</i>
SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
server config, virtual host, directory, .htaccess
AuthConfig
(E)
mod_ssl

Equivalent to SSLCipherSuite, but for the proxy connection. Please refer to [SSLCipherSuite](#) for additional information.



SSL Proxy Engine

SSL Proxy Engine Operation Switch
SSLProxyEngine on off
SSLProxyEngine off
server config, virtual host
(E)
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.

```
<VirtualHost _default_:443>  
SSLProxyEngine on  
...  
</VirtualHost>
```



SSLProxyMachineCertificateFile

File of concatenated PEM-encoded client certificates and keys to be used by the proxy
SSLProxyMachineCertificateFile <i>filename</i>
server config
Not applicable
(E)
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

```
SSLProxyMachineCertificateFile  
/usr/local/apache2/conf/ssl.crt/proxy.pem
```



SSLProxyMachineCertificatePath

Directory of PEM-encoded client certificates and keys to be used by the proxy
SSLProxyMachineCertificatePath <i>directory</i>
server config
Not applicable
(E)
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

```
SSLProxyMachineCertificatePath  
/usr/local/apache2/conf/proxy.crt/
```



SSLProxyProtocol

Configure usable SSL protocol flavors for proxy usage
SSLProxyProtocol [+ -] <i>protocol</i> ...
SSLProxyProtocol all
server config, virtual host
Options
(E)
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.



Type of remote server Certificate verification
SSLProxyVerify <i>level</i>
SSLProxyVerify none
server config, virtual host, directory, .htaccess
AuthConfig
(E)
mod_ssl

When a proxy is configured to forward requests to a remote SSL server, this directive can be used to configure certificate verification of the remote server. 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 by the proxy. 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.

Note that even when certificate verification is enabled, [mod_ssl](#) does **not** check whether the commonName (hostname) attribute of the server certificate matches the hostname used to connect to the server. In other words, the proxy does not guarantee that the SSL connection to the backend server is "secure" beyond the fact that the certificate is signed by one of the CAs configured using the [SSLProxyCACertificatePath](#) and/or [SSLProxyCACertificateFile](#) directives.

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 **nonerequire** 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.)

```
SSLProxyVerify require
```



SSLProxyVerifyDepth

Maximum depth of CA Certificates in Remote Server Certificate verification
SSLProxyVerifyDepth <i>number</i>
SSLProxyVerifyDepth 1
server config, virtual host, directory, .htaccess
AuthConfig
(E)
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.

```
SSLProxyVerifyDepth 10
```



Pseudo Random Number Generator (PRNG) seeding source
SSLRandomSeed <i>context source [bytes]</i>
server config
(E)
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 `0` is given to `/path/to/source` as the first argument). Use this

especially at startup time, for instance with an available `/dev/random` and/or `/dev/urandom` devices (which usually exist on modern Unix derivatives like FreeBSD and Linux).

But be careful: Usually `/dev/random` provides only as much entropy data as it actually has, i.e. when you request 512 bytes of entropy, but the device currently has only 100 bytes available two things can happen: On some platforms you receive only the 100 bytes while on other platforms the read blocks until enough bytes are available (which can take a long time). Here using an existing `/dev/urandom` is better, because it never blocks and actually gives the amount of requested data. The drawback is just that the quality of the received data may not be the best.

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 its client program with the exec `:/path/to/program/ variant` (see below) or use `egd:/path/to/egd-socket` (see below).

- `exec:/path/to/program`
This variant uses an external executable `/path/to/program` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of its `stdout` contents form the entropy. When *bytes* is not specified, the entirety of the data produced on `stdout` form the entropy. Use this only at startup time when you need a very strong seeding with the help of an external program (for instance as in the example above with the `truerand` utility you can find in the `mod_ssl` distribution which is based on the AT&T *truerand* library). Using this in the connection context slows down the server too dramatically, of course. So usually you should avoid using external programs in that context.

- `egd:/path/to/egd-socket` (Unix only)
This variant uses the Unix domain socket of the external Entropy Gathering Daemon (EGD) (see <http://www.lothar.com/tech/crypto/>) to seed the PRNG. Use this if no random device exists on your platform.

```
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
```



Allow access only when an arbitrarily complex boolean expression is true
SSLRequire <i>expression</i>
directory, .htaccess
AuthConfig
(E)
mod_ssl

This directive specifies a general access requirement which has to be fulfilled in order to allow access. It is a very powerful directive because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks.

The implementation of SSLRequire is not thread safe. Using SSLRequire inside .htaccess files on a threaded [MPM](#) may cause random crashes.

The *expression* must match the following syntax (given as a BNF grammar notation):

```

expr      ::= "true" | "false"
           | "!" expr
           | expr "&&" expr
           | expr "||" expr
           | "(" expr ")"
           | comp

comp      ::= word "==" word | word "eq" word
           | word "!=" word | word "ne" word
           | word "<" word | word "lt" word
           | word "<=" word | word "le" word
           | word ">" word | word "gt" word
           | word ">=" word | word "ge" word

```

```

        | word "in" "{" wordlist "}"
        | word "in" "OID(" word ")"
        | word "=~" regex
        | word "!~" regex

wordlist ::= word
          | wordlist ", " word

word     ::= digit
          | cstring
          | variable
          | function

digit    ::= [0-9]+
cstring  ::= "... "
variable ::= "%{" varname "}"
function ::= funcname "(" funcargs ")"

```

while for varname any variable from [Table 3](#) can be used. Finally for funcname the following functions are available:

- `file(filename)`

This function takes one string argument and expands to the contents of the file. This is especially useful for matching this contents against a regular expression, etc.

Notice that *expression* is first parsed into an internal machine representation and then evaluated in a second step. Actually, in Global and Per-Server Class context *expression* is parsed at startup time and at runtime only the machine representation is executed. For Per-Directory context this is different: here *expression* has to be parsed and immediately executed for every request.

```

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]+$/

```

OID() function expects to find zero or more instances of the given OID in the client certificate, and compares the left-hand side string against the value of matching OID attributes. Every matching OID is checked, until a match is found.

Standard CGI/1.0 and Apache variables:

HTTP_USER_AGENT	PATH_INFO	AUTH_TYPE
HTTP_REFERER	QUERY_STRING	SERVER_SO
HTTP_COOKIE	REMOTE_HOST	API_VERSION
HTTP_FORWARDED	REMOTE_IDENT	TIME_YEAR
HTTP_HOST	IS_SUBREQ	TIME_MON
HTTP_PROXY_CONNECTION	DOCUMENT_ROOT	TIME_DAY
HTTP_ACCEPT	SERVER_ADMIN	TIME_HOUR
HTTP:headername	SERVER_NAME	TIME_MIN
THE_REQUEST	SERVER_PORT	TIME_SEC
REQUEST_METHOD	SERVER_PROTOCOL	TIME_WDAY
REQUEST_SCHEME	REMOTE_ADDR	TIME
REQUEST_URI	REMOTE_USER	ENV:varia
REQUEST_FILENAME		

SSL-related variables:

HTTPS	SSL_CLIENT_M_VERSION	SSL_SERVI
	SSL_CLIENT_M_SERIAL	SSL_SERVI
SSL_PROTOCOL	SSL_CLIENT_V_START	SSL_SERVI
SSL_SESSION_ID	SSL_CLIENT_V_END	SSL_SERVI
SSL_CIPHER	SSL_CLIENT_S_DN	SSL_SERVI
SSL_CIPHER_EXPORT	SSL_CLIENT_S_DN_C	SSL_SERVI
SSL_CIPHER_ALGKEYSIZE	SSL_CLIENT_S_DN_ST	SSL_SERVI
SSL_CIPHER_USEKEYSIZE	SSL_CLIENT_S_DN_L	SSL_SERVI

SSL_VERSION_LIBRARY	SSL_CLIENT_S_DN_O	SSL_SERVI
SSL_VERSION_INTERFACE	SSL_CLIENT_S_DN_OU	SSL_SERVI
	SSL_CLIENT_S_DN_CN	SSL_SERVI
	SSL_CLIENT_S_DN_T	SSL_SERVI
	SSL_CLIENT_S_DN_I	SSL_SERVI
	SSL_CLIENT_S_DN_G	SSL_SERVI
	SSL_CLIENT_S_DN_S	SSL_SERVI
	SSL_CLIENT_S_DN_D	SSL_SERVI
	SSL_CLIENT_S_DN_UID	SSL_SERVI
	SSL_CLIENT_S_DN_Email	SSL_SERVI
	SSL_CLIENT_I_DN	SSL_SERVI
	SSL_CLIENT_I_DN_C	SSL_SERVI
	SSL_CLIENT_I_DN_ST	SSL_SERVI
	SSL_CLIENT_I_DN_L	SSL_SERVI
	SSL_CLIENT_I_DN_O	SSL_SERVI
	SSL_CLIENT_I_DN_OU	SSL_SERVI
	SSL_CLIENT_I_DN_CN	SSL_SERVI
	SSL_CLIENT_I_DN_T	SSL_SERVI
	SSL_CLIENT_I_DN_I	SSL_SERVI
	SSL_CLIENT_I_DN_G	SSL_SERVI
	SSL_CLIENT_I_DN_S	SSL_SERVI
	SSL_CLIENT_I_DN_D	SSL_SERVI
	SSL_CLIENT_I_DN_UID	SSL_SERVI
	SSL_CLIENT_I_DN_Email	SSL_SERVI
	SSL_CLIENT_A_SIG	SSL_SERVI
	SSL_CLIENT_A_KEY	SSL_SERVI
	SSL_CLIENT_CERT	SSL_SERVI
	SSL_CLIENT_CERT_CHAIN_n	
	SSL_CLIENT_VERIFY	



Deny access when SSL is not used for the HTTP request
SSLRequireSSL
directory, .htaccess
AuthConfig
(E)
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.

SSLRequireSSL



Type of the global/inter-process SSL Session Cache
SSLSessionCache <i>type</i>
SSLSessionCache none
server config
(E)
mod_ssl

This configures the storage type of the global/inter-process SSL Session Cache. This cache is an optional facility which speeds up parallel request processing. For requests to the same server process (via HTTP keep-alive), OpenSSL already caches the SSL session information locally. But because modern clients request inlined images and other data via parallel requests (usually up to four parallel requests are common) those requests are served by *different* pre-forked server processes. Here an inter-process cache helps to avoid unnecessary session handshakes.

The following four storage *types* are currently supported:

- none
This disables the global/inter-process Session Cache. This will incur a noticeable speed penalty and may cause problems if using certain browsers, particularly if client certificates are enabled. This setting is not recommended.
- nonenotnull
This disables any global/inter-process Session Cache. However it does force OpenSSL to send a non-null session ID to accommodate buggy clients that require one.
- dbm:/path/to/datafile
This makes use of a DBM hashfile on the local disk to synchronize the local OpenSSL memory caches of the server

processes. This session cache may suffer reliability issues under high load.

- `shm:/path/to/datafile[(size)]`
This makes use of a high-performance cyclic buffer (approx. *size* bytes in size) inside a shared memory segment in RAM (established via `/path/to/datafile`) to synchronize the local OpenSSL memory caches of the server processes. This is the recommended session cache.
- `dc:UNIX:/path/to/socket`
This makes use of the [distcache](#) distributed session caching libraries. The argument should specify the location of the server or proxy to be used using the distcache address syntax; for example, `UNIX:/path/to/socket` specifies a UNIX domain socket (typically a local `dc_client` proxy);
`IP:server.example.com:9001` specifies an IP address.

```
SSLSessionCache
```

```
dbm:/usr/local/apache/logs/ssl_gcach_data
```

```
SSLSessionCache
```

```
shm:/usr/local/apache/logs/ssl_gcach_data(512000)
```



Number of seconds before an SSL session expires in the Session Cache
<code>SSLSessionCacheTimeout</code> <i>seconds</i>
<code>SSLSessionCacheTimeout</code> 300
server config, virtual host
(E)
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.

```
SSLSessionCacheTimeout 600
```



Variable name to determine user name
SSLUserName <i>varname</i>
server config, directory, .htaccess
AuthConfig
(E)
mod_ssl
Apache 2.0.51

This directive sets the "user" field in the Apache request object. This is used by lower modules to identify the user with a character string. In particular, this may cause the environment variable REMOTE_USER to be set. The *varname* can be any of the [SSL environment variables](#).

Note that this directive has no effect if the FakeBasic option is used (see [SSLOptions](#)).

```
SSLUserName SSL_CLIENT_S_DN_CN
```



Type of Client Certificate verification
SSLVerifyClient <i>level</i>
SSLVerifyClient none
server config, virtual host, directory, .htaccess
AuthConfig
(E)
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 **nonerequire** 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.)

```
SSLVerifyClient require
```



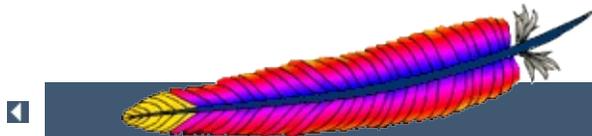
SSLVerifyDepth

Maximum depth of CA Certificates in Client Certificate verification
SSLVerifyDepth <i>number</i>
SSLVerifyDepth 1
server config, virtual host, directory, .htaccess
AuthConfig
(E)
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.

```
SSLVerifyDepth 10
```



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_status

Web
(B)
status_module
mod_status.c

The Status module allows a server administrator to find out how well their server is performing. A HTML page is presented that gives the current server statistics in an easily readable form. If required this page can be made to automatically refresh (given a compatible browser). Another page gives a simple machine-readable list of the current server state.

The details given are:

- The number of worker serving requests
- The number of idle worker
- The status of each worker, the number of requests that worker has performed and the total number of bytes served by the worker (*)
- A total number of accesses and byte count served (*)
- The time the server was started/restarted and the time it has been running for
- Averages giving the number of requests per second, the number of bytes served per second and the average number of bytes per request (*)
- The current percentage CPU used by each worker and in total by Apache (*)
- The current hosts and requests being processed (*)

A compile-time option must be used to display the details marked "

(*)" as the instrumentation required for obtaining these statistics does not exist within standard Apache.



Enabling Status Support

To enable status reports only for browsers from the foo.com domain add this code to your `httpd.conf` configuration file

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

Order Deny,Allow
Deny from all
Allow from .foo.com
</Location>
```

You can now access server statistics by using a Web browser to access the page `http://your.server.name/server-status`



Automatic updates

You can get the status page to update itself automatically if you have a browser that supports "refresh". Access the page `http://your.server.name/server-status?refresh=N` to refresh the page every N seconds.



Machine-Readable Status File

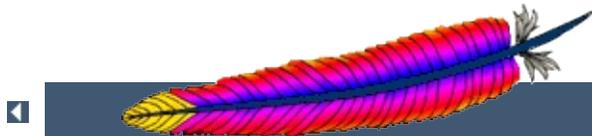
A machine-readable version of the status file is available by accessing the page `http://your.server.name/server-status?auto`. This is useful when automatically run, see the Perl program in the `/support` directory of Apache, `log_server_status`.

It should be noted that if `mod_status` is compiled into the server, its handler capability is available in *all* configuration files, including *per-directory* files (`.htaccess`). This may have security-related ramifications for your site.



Keep track of extended status information for each request
ExtendedStatus On Off
ExtendedStatus Off
server config
(B)
mod_status
ExtendedStatus is only available in Apache 1.3.2

This setting applies to the entire server, and cannot be enabled or disabled on a virtualhost-by-virtualhost basis. The collection of extended status information can slow down the server.



Apache HTTP Server 2.2

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

| | | 2006129 |

Apache mod_suexec

webCGISSI
(E)
suexec_module
mod_suexec.c
Apache 2.0

[suexecCGI](#)

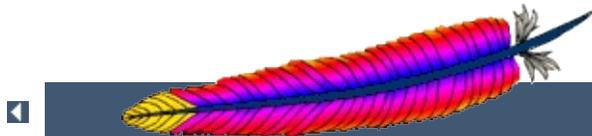


SuexecUserGroup

CGI
SuexecUserGroup <i>User Group</i>
server config, virtual host
(E)
mod_suexec
Apache 2.0

SuexecUserGroupCGICGIUserApache1.3VirtualHostsUser
Group

SuexecUserGroup nobody nogroup



||||

◀ Apache HTTP Server 2.2

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

|| < > | ??? |

Apache mod_unique_id

(E)
unique_id_module
mod_unique_id.c

This module provides a magic token for each request which is guaranteed to be unique across "all" requests under very specific conditions. The unique identifier is even unique across multiple machines in a properly configured cluster of machines. The environment variable `UNIQUE_ID` is set to the identifier for each request. Unique identifiers are useful for various reasons which are beyond the scope of this document.



First a brief recap of how the Apache server works on Unix machines. This feature currently isn't supported on Windows NT. On Unix machines, Apache creates several children, the children process requests one at a time. Each child can serve multiple requests in its lifetime. For the purpose of this discussion, the children don't share any data with each other. We'll refer to the children as *httpd processes*.

Your website has one or more machines under your administrative control, together we'll call them a cluster of machines. Each machine can possibly run multiple instances of Apache. All of these collectively are considered "the universe", and with certain assumptions we'll show that in this universe we can generate unique identifiers for each request, without extensive communication between machines in the cluster.

The machines in your cluster should satisfy these requirements. (Even if you have only one machine you should synchronize its clock with NTP.)

- The machines' times are synchronized via NTP or other network time protocol.
- The machines' hostnames all differ, such that the module can do a hostname lookup on the hostname and receive a different IP address for each machine in the cluster.

As far as operating system assumptions go, we assume that pids (process ids) fit in 32-bits. If the operating system uses more than 32-bits for a pid, the fix is trivial but must be performed in the code.

Given those assumptions, at a single point in time we can identify any httpd process on any machine in the cluster from all other httpd processes. The machine's IP address and the pid of the httpd process are sufficient to do this. So in order to generate unique identifiers for

requests we need only distinguish between different points in time.

To distinguish time we will use a Unix timestamp (seconds since January 1, 1970 UTC), and a 16-bit counter. The timestamp has only one second granularity, so the counter is used to represent up to 65536 values during a single second. The quadruple (*ip_addr*, *pid*, *time_stamp*, *counter*) is sufficient to enumerate 65536 requests per second per httpd process. There are issues however with pid reuse over time, and the counter is used to alleviate this issue.

When an httpd child is created, the counter is initialized with (current microseconds divided by 10) modulo 65536 (this formula was chosen to eliminate some variance problems with the low order bits of the microsecond timers on some systems). When a unique identifier is generated, the time stamp used is the time the request arrived at the web server. The counter is incremented every time an identifier is generated (and allowed to roll over).

The kernel generates a pid for each process as it forks the process, and pids are allowed to roll over (they're 16-bits on many Unixes, but newer systems have expanded to 32-bits). So over time the same pid will be reused. However unless it is reused within the same second, it does not destroy the uniqueness of our quadruple. That is, we assume the system does not spawn 65536 processes in a one second interval (it may even be 32768 processes on some Unixes, but even this isn't likely to happen).

Suppose that time repeats itself for some reason. That is, suppose that the system's clock is screwed up and it revisits a past time (or it is too far forward, is reset correctly, and then revisits the future time). In this case we can easily show that we can get pid and time stamp reuse. The choice of initializer for the counter is intended to help defeat this. Note that we really want a random number to initialize the counter, but there aren't any readily available numbers on most systems (*i.e.*, you can't use `rand()` because you need to seed the

generator, and can't seed it with the time because time, at least at one second resolution, has repeated itself). This is not a perfect defense.

How good a defense is it? Suppose that one of your machines serves at most 500 requests per second (which is a very reasonable upper bound at this writing, because systems generally do more than just shovel out static files). To do that it will require a number of children which depends on how many concurrent clients you have. But we'll be pessimistic and suppose that a single child is able to serve 500 requests per second. There are 1000 possible starting counter values such that two sequences of 500 requests overlap. So there is a 1.5% chance that if time (at one second resolution) repeats itself this child will repeat a counter value, and uniqueness will be broken. This was a very pessimistic example, and with real world values it's even less likely to occur. If your system is such that it's still likely to occur, then perhaps you should make the counter 32 bits (by editing the code).

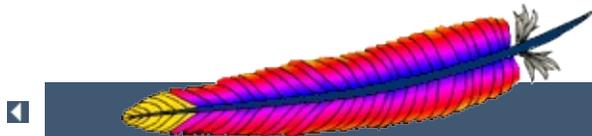
You may be concerned about the clock being "set back" during summer daylight savings. However this isn't an issue because the times used here are UTC, which "always" go forward. Note that x86 based Unixes may need proper configuration for this to be true -- they should be configured to assume that the motherboard clock is on UTC and compensate appropriately. But even still, if you're running NTP then your UTC time will be correct very shortly after reboot.

UNIQUE_ID environment variable is constructed by encoding the 112-bit (32-bit IP address, 32 bit pid, 32 bit time stamp, 16 bit counter) quadruple using the alphabet [A-Za-z0-9@-] in a manner similar to MIME base64 encoding, producing 19 characters. The MIME base64 alphabet is actually [A-Za-z0-9+/] however +/ need to be specially encoded in URLs, which makes them less desirable. All values are encoded in network byte ordering so that the encoding is comparable across architectures of different byte ordering. The actual ordering of the encoding is: time stamp, IP address, pid, counter. This ordering has a purpose, but it should be emphasized that applications

should not dissect the encoding. Applications should treat the entire encoded `UNIQUE_ID` as an opaque token, which can be compared against other `UNIQUE_IDs` for equality only.

The ordering was chosen such that it's possible to change the encoding in the future without worrying about collision with an existing database of `UNIQUE_IDs`. The new encodings should also keep the time stamp as the first element, and can otherwise use the same alphabet and bit length. Since the time stamps are essentially an increasing sequence, it's sufficient to have a *flag second* in which all machines in the cluster stop serving and request, and stop using the old encoding format. Afterwards they can resume requests and begin issuing the new encodings.

This we believe is a relatively portable solution to this problem. It can be extended to multithreaded systems like Windows NT, and can grow with future needs. The identifiers generated have essentially an infinite life-time because future identifiers can be made longer as required. Essentially no communication is required between machines in the cluster (only NTP synchronization is required, which is low overhead), and no communication between `httpd` processes is required (the communication is implicit in the `pid` value assigned by the kernel). In very specific situations the identifier can be shortened, but more information needs to be assumed (for example the 32-bit IP address is overkill for any site, but there is no portable shorter replacement for it).



Apache HTTP Server 2.2

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

| | | 2006129 |

Apache mod_userdir

("/~username")
(B)
userdir_module
mod_userdir.c

`http://example.com/~user/`



UserDir <i>directory-filename</i>
server config, virtual host
(B)
mod_userdir

UserDir *Directory-filename*

- disabled enabled()
- disabled(enabled)
- enabled disabled disabled

Userdir enableddisabled
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	http://www.foo.com/users/bob/one/two.html
UserDir http://www.foo.com/*/usr	http://www.foo.com/bob/usr/one/two.html
UserDir http://www.foo.com/~*/	http://www.foo.com/~bob/one/two.html

" **UserDir** ./"" /~root" /"" **UserDir**

disabled root" Directory

UserDir

```
UserDir disabled  
UserDir enabled user1 user2 user3
```

UserDir

```
UserDir enabled  
UserDir disabled user4 user5 user6
```

(alternative)

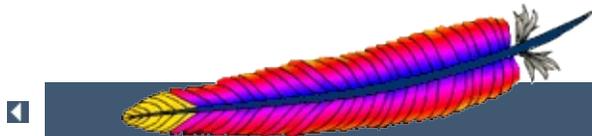
```
Userdir public_html /usr/web http://www.foo.com/
```

```
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

2.1.4 UserDir" UserDir public_html"

-



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_usertrack

Session(Cookie)
(E)
usertrack_module
mod_usertrack.c

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 that m.

True, Netscape claims that the correct format NOW is for
four digit dates do in fact work... for Netscape 4.x (4.0
is. However, 3.x and below do NOT accept them. It seems
originally had a 2-digit standard, and then with all of
probably a few complaints, changed to a four digit date.
Fortunately, 4.x also understands the 2-digit format, so
ensure that your expiration date is legible to the client.
use 2-digit dates.

However, this does not limit expiration dates to the year
an expiration year of "13", for example, it is interpreted
1913! In fact, you can use an expiration year of up to
understood as "2037" by both MSIE and Netscape versions
about versions previous to those). Not sure why Netscape
particular year as its cut-off point, but my guess is related
to UNIX's 2038 problem. Netscape/MSIE 4.x seem to be able to
2-digit years beyond that, at least until "50" for sure
understand up until about "70", but not for sure).

Summary: Mozilla 3.x and up understands two digit dates
(2037). Mozilla 4.x understands up until at least "50"
form, but also understands 4-digit years, which can probably
9999. Your best bet for sending a long-life cookie is
time late in the year "37".



The domain to which the tracking cookie applies
CookieDomain <i>domain</i>
server config, virtual host, directory, .htaccess
FileInfo
(E)
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.



Expiry time for the tracking cookie
CookieExpires <i>expiry-period</i>
server config, virtual host, directory, .htaccess
FileInfo
(E)
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.



Name of the tracking cookie
CookieName <i>token</i>
CookieName Apache
server config, virtual host, directory, .htaccess
FileInfo
(E)
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 "-".



Format of the cookie header field
CookieStyle <i>Netscape Cookie Cookie2 RFC2109 RFC2965</i>
CookieStyle Netscape
server config, virtual host, directory, .htaccess
FileInfo
(E)
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.
- **CookieRFC2109**, which is the syntax that superseded the Netscape syntax.
- **Cookie2RFC2965**, which is the most current cookie syntax.

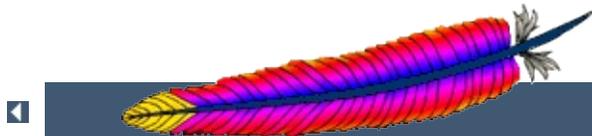
Not all clients can understand all of these formats. but you should use the newest one that is generally acceptable to your users' browsers.



CookieTracking

Enables tracking cookie
CookieTracking on off
CookieTracking off
server config, virtual host, directory, .htaccess
FileInfo
(E)
mod_usertrack

When the user track module is compiled in, and "CookieTracking on" is set, Apache will start sending 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, compiling mod_usertrack will not activate cookies.



Apache HTTP Server 2.2

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

| | < > | ??? |

Apache mod_version

(E)
version_module
mod_version.c
Apache 2.0.56

This module is designed for the use in test suites and large networks which have to deal with different httpd versions and different configurations. It provides a new container -- [<IfVersion>](#), which allows a flexible version checking including numeric comparisons and regular expressions.

```
<IfVersion 2.1.0>
  # current httpd version is exactly 2.1.0
</IfVersion>

<IfVersion >= 2.2>
  # use really new features :- )
</IfVersion>
```

See below for further possibilities.



contains version dependent configuration
<code><IfVersion [[!]operator] version> ... </IfVersion></code>
server config, virtual host, directory, .htaccess
All
(E)
mod_version

`<IfVersion>` section encloses configuration directives which are executed only if the [httpd](#) version matches the desired criteria. For normal (numeric) comparisons the *version* argument has the format *major[.minor[.patch]]*, e.g. 2.1.02.2. *minor* and *patch* are optional. If these numbers are omitted, they are assumed to be zero. The following numerical *operators* are possible:

<i>operator</i>	<i>description</i>
===	httpd version is equal
>	httpd version is greater than
>=	httpd version is greater or equal
<	httpd version is less than
<=	httpd version is less or equal

```
<IfVersion >= 2.1>
  # this happens only in versions greater or
  # equal 2.1.0.
</IfVersion>
```

Besides the numerical comparison it is possible to match a [regular expression](#) against the httpd version. There are two ways to write it:

<i>operator</i>	<i>description</i>
-----------------	--------------------

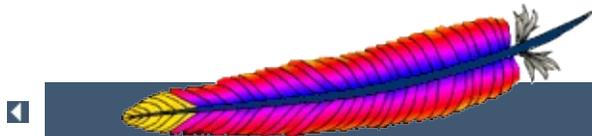
===	<i>version</i> has the form <i>/regex/</i>
~	<i>version</i> has the form <i>regex</i>

```
<IfVersion = /^2.1.[01234]$/>  
  # e.g. workaround for buggy versions  
</IfVersion>
```

In order to reverse the meaning, all operators can be preceded by an exclamation mark (!):

```
<IfVersion !~ ^2.1.[01234]$/>  
  # not for those versions  
</IfVersion>
```

If the *operator* is omitted, it is assumed to be =.



Apache HTTP Server 2.2

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

| | | 2006129 |

Apache mod_vhost_alias

(E)
vhost_alias_module
mod_vhost_alias.c

HTTP/IP" Host:"

```
mod aliasmod userdirURI mod vhost alias  
/cgi-bin/script.pl /usr/local/apache2/cgi-  
bin/script.pl
```

```
ScriptAlias /cgi-bin/  
/usr/local/apache2/cgi-bin/  
VirtualScriptAlias /never/foand/%0/cgi-  
bin/
```



("%name")(UseCanonicalName)""IP printf

%%	(%)
%p	
%N.M	()

NMname Nname MN M"0" M

0	name
1	
2	
-1	
-2	
2+	
-2+	
1+ -1+	0

NM



```
UseCanonicalName Off
VirtualDocumentRoot /usr/local/apache/vhosts/%0
```

```
http://www.example.com/directory/file.html
/usr/local/apache/vhosts/www.example.com/directory/file.html

vhosts
```

```
UseCanonicalName Off
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

```
http://www.domain.example.com/directory/file.html
/usr/local/apache/vhosts/example.com/d/o/m/domain/directory

name(hashing)
```

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

```
/usr/local/apache/vhosts/example.com/n/i/a/domain/directory
```

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

```
/usr/local/apache/vhosts/example.com/d/o/m/ain/directory
```

IP

```
UseCanonicalName DNS
VirtualDocumentRootIP
/usr/local/apache/vhosts/%1/%2/%3/%4/docs
VirtualScriptAliasIP
/usr/local/apache/vhosts/%1/%2/%3/%4/cgi-bin
```

```
http://www.domain.example.com/directory/file.html
/usr/local/apache/vhosts/10/20/30/40/docs/directory/fi.
    www.domain.example.comIP10.20.30.40
http://www.domain.example.com/cgi-bin/script.pl
/usr/local/apache/vhosts/10/20/30/40/cgi-
bin/script.pl
```

VirtualDocumentRoot(.) %

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%2.0.%3.0
```

```
http://www.domain.example.com/directory/file.html
/usr/local/apache/vhosts/domain.example/directory/file
```

LogFormat%V%A



VIRTUALDOCUMENTROOT

VirtualDocumentRoot <i>interpolated-directory</i> none
VirtualDocumentRoot none
server config, virtual host
(E)
mod_vhost_alias

VirtualDocumentRootApache *interpolated-directory*

DocumentRoot *interpolated-directory*none

VirtualDocumentRoot VirtualDocumentRootIP



VIRTUALDOCUMENTROOT

IP
VirtualDocumentRootIP <i>interpolated-directory</i> none
VirtualDocumentRootIP none
server config, virtual host
(E)
mod_vhost_alias

VirtualDocumentRootIP VirtualDocumentRootIP



VirtualScriptAlias

CGI
VirtualScriptAlias <i>interpolated-directory</i> none
VirtualScriptAlias none
server config, virtual host
(E)
mod_vhost_alias

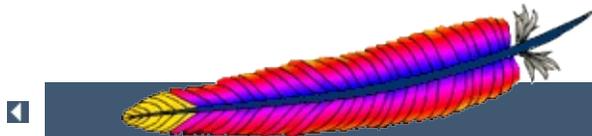
VirtualScriptAliasApacheCGI VirtualDocumentRoot
/cgi-bin/URI" ScriptAlias /cgi-bin/"



VirtualScriptAlias

IPCGI
VirtualScriptAliasIP <i>interpolated-directory</i> none
VirtualScriptAliasIP none
server config, virtual host
(E)
mod_vhost_alias

VirtualScriptAliasIP VirtualScriptAliasIP



Apache HTTP Server 2.2

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

| | < > | ??? |

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`

`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.

```
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 success */
    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 GI
char *protocol;      /* Protocol, as given to us, or I
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 file:  
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_popen`, which also arranges for the underlying file descriptor to be closed before any child processes, such as for CGI scripts, are executed), 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_pcallloc` 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`,

```
...  
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_pfclose` and `ap_pcclosef`. (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 `fclose` 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 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 Core

One of the design goals for this server was to maintain external compatibility with the NCSA 1.3 server --- that is, to read the same configuration files, to process all the directives therein correctly, and in general to be a drop-in replacement for NCSA. On the other hand, another design goal was to move as much of the server's functionality into modules which have as little as possible to do with the monolithic server core. The only way to reconcile these goals is to move the handling of most commands from the central server into the modules.

However, just giving the modules command tables is not enough to divorce them completely from the server core. The server has to remember the commands in order to act on them later. That involves maintaining data which is private to the modules, and which can be either per-server, or per-directory. Most things are per-directory, including in particular access control and authorization information, but also information on how to determine file types from suffixes, which can be modified by `AddType` and `DefaultType` directives, and so forth. In general, the governing philosophy is that anything which *can* be made configurable by directory should be; per-server information is generally used in the standard set of modules for information like `Aliases` and `Redirects` which come into play before the request is tied to a particular place in the underlying file system.

Another requirement for emulating the NCSA server is being able to handle the per-directory configuration files, generally called `.htaccess` files, though even in the NCSA server they can contain directives which have nothing at all to do with access control. Accordingly, after URI -> filename translation, but before performing any other phase, the server walks down the directory hierarchy of the underlying filesystem, following the translated pathname, to read any `.htaccess` files which might be present. The information which is read in then has to be *merged* with the applicable information from the

server's own config files (either from the [<Directory>](#) sections in `access.conf`, or from defaults in `srmd.conf`, which actually behaves for most purposes almost exactly like `<Directory />`).

Finally, after having served a request which involved reading `.htaccess` files, we need to discard the storage allocated for handling them. That is solved the same way it is solved wherever else similar problems come up, by tying those structures to the per-transaction resource pool.

Per-directory configuration structures

Let's look out how all of this plays out in `mod_mime.c`, which defines the file typing handler which emulates the NCSA server's behavior of determining file types from suffixes. What we'll be looking at, here, is the code which implements the [AddType](#) and [AddEncoding](#) commands. These commands can appear in `.htaccess` files, so they must be handled in the module's private per-directory data, which in fact, consists of two separate tables for MIME types and encoding information, and is declared as follows:

```
typedef struct {
    table *forced_types;      /* Additional AddType'd :
    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 `srmd.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 (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 (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).

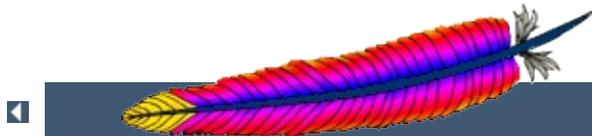
The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the `alias` module, which also indicates how a syntax error can be returned

(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;
}
```



||||



Apache HTTP Server 2.2

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

||< >|???

Debugging Memory Allocation in APR

The allocation mechanisms within APR have a number of debugging modes that can be used to assist in finding memory problems. This document describes the modes available and gives instructions on activating them.



Allocation Debugging - `ALLOC_DEBUG`

Debugging support: Define this to enable code which helps detect re-use of `free()`d memory and other such nonsense.

The theory is simple. The `FILL_BYTE` (`0xa5`) is written over all `malloc`'d memory as we receive it, and is written over everything that we free up during a `clear_pool`. We check that blocks on the free list always have the `FILL_BYTE` in them, and we check during `palloc()` that the bytes still have `FILL_BYTE` in them. If you ever see garbage URLs or whatnot containing lots of `0xa5`s then you know something used data that's been freed or uninitialized.

Malloc Support - `ALLOC_USE_MALLOC`

If defined all allocations will be done with `malloc()` and `free()`d appropriately at the end.

This is intended to be used with something like Electric Fence or Purify to help detect memory problems. Note that if you're using efence then you should also add in `ALLOC_DEBUG`. But don't add in `ALLOC_DEBUG` if you're using Purify because `ALLOC_DEBUG` would hide all the uninitialized read errors that Purify can diagnose.

Pool Debugging - `POOL_DEBUG`

This is intended to detect cases where the wrong pool is used when assigning data to an object in another pool.

In particular, it causes the `table_{set, add, merge}` routines to check that their arguments are safe for the `apr_table_t` they're

being placed in. It currently only works with the unix multiprocess model, but could be extended to others.

Table Debugging - MAKE_TABLE_PROFILE

Provide diagnostic information about `make_table()` calls which are possibly too small.

This requires a recent gcc which supports `__builtin_return_address()`. The `error_log` output will be a message such as:

```
table_push: apr_table_t created by 0x804d874 hit  
limit of 10
```

Use `l *0x804d874` to find the source that corresponds to. It indicates that a `apr_table_t` allocated by a call at that address has possibly too small an initial `apr_table_t` size guess.

Allocation Statistics - ALLOC_STATS

Provide some statistics on the cost of allocations.

This requires a bit of an understanding of how `alloc.c` works.



Not all the options outlined above can be activated at the same time. the following table gives more information.

	ALLOC DEBUG	ALLOC USE MALLOC	POOL DEBUG	MAKE TABLE PROFILE	ALLOC STATS
ALLOC DEBUG	-	No	Yes	Yes	Yes
ALLOC USE MALLOC	No	-	No	No	No
POOL DEBUG	Yes	No	-	Yes	Yes
MAKE TABLE PROFILE	Yes	No	Yes	-	Yes
ALLOC STATS	Yes	No	Yes	Yes	-

Additionally the debugging options are not suitable for multi-threaded versions of the server. When trying to debug with these options the server should be started in single process mode.



Remaining Debugging Options

The various options for debugging memory are now enabled in the `apr_general.h` header file in APR. The various options are enabled by uncommenting the define for the option you wish to use. The section of the code currently looks like this (*contained in `srclib/apr/include/apr_pools.h`*)

```
/*
#define ALLOC_DEBUG
#define POOL_DEBUG
#define ALLOC_USE_MALLOC
#define MAKE_TABLE_PROFILE
#define ALLOC_STATS
*/

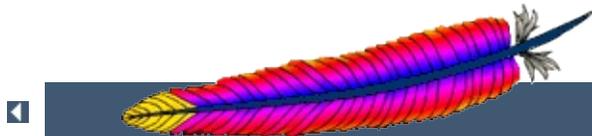
typedef struct ap_pool_t {
    union block_hdr *first;
    union block_hdr *last;
    struct cleanup *cleanups;
    struct process_chain *subprocesses;
    struct ap_pool_t *sub_pools;
    struct ap_pool_t *sub_next;
    struct ap_pool_t *sub_prev;
    struct ap_pool_t *parent;
    char *free_first_avail;
#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.



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Documenting Apache 2.0

Apache 2.0 uses [Doxygen](#) to document the APIs and global variables in the 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
```

`deffunc` is not always necessary. DoxyGen does not have a full parser in it, so any prototype that use a macro in the return type declaration is too complex for scandoc. Those functions require a `deffunc`. An example (using `>` rather than `>`):

```
/**
 * return the final element of the pathname
 * @param pathname The path to get the final
element of
 * @return the final element of the path
 * @tip Examples:
 * <pre>
 * "/foo/bar/gum" -&gt; "gum"
 * "/foo/bar/gum/" -&gt; ""
 * "gum" -&gt; "gum"
 * "wi\\n32\\stuff" -&gt; "stuff"
 * </pre>
 * @deffunc const char *
ap_filename_of_pathname(const char *pathname)
 */
```

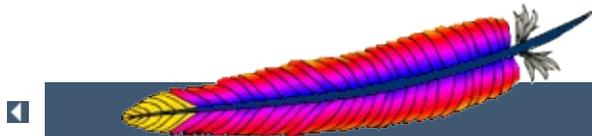
At the top of the header file, always include:

```
/**
```

```
* @package Name of library header  
*/
```

Doxygen uses a new HTML file for each package. The HTML files are named {Name_of_library_header}.html, so try to be concise with your names.

For a further discussion of the possibilities please refer to [the Doxygen site](#).



Apache HTTP Server 2.2

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

| | < > | ??? |

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, OKDECLINED 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, APR_HOOK_MIDDLE);
}

module MODULE_VAR_EXPORT my_module =
{
    ...
    my_register_hooks /* register hooks */
};
```

Controlling hook calling order

In the example above, we didn't use the three arguments in the hook registration function that control calling order. There are two

mechanisms for doing this. The first, rather crude, method, allows us to specify roughly where the hook is run relative to other modules. The final argument control this. There are three possible values: APR_HOOK_FIRST, APR_HOOK_MIDDLE, APR_HOOK_LAST.

All modules using any particular value may be run in any order relative to each other, but, of course, all modules using APR_HOOK_FIRST will be run before APR_HOOK_MIDDLE which are before APR_HOOK_LAST. Modules that don't care when they are run should use APR_HOOK_MIDDLE. *(I spaced these out so people could do stuff like APR_HOOK_FIRST - 2 to get in slightly earlier, but is this wise? - Ben)*

Note that there are two more values, APR_HOOK_REALLY_FIRST, APR_HOOK_REALLY_LAST. These should only be used by the hook exporter.

The other method allows finer control. When a module knows that it must be run before (or after) some other modules, it can specify them by name. The second (third) argument is a NULL-terminated array of strings consisting of the names of modules that must be run before (after) the current module. For example, suppose we want "mod_xyz.c" and "mod_abc.c" to run before we do, then we'd hook as follows:

```
static void register_hooks()
{
    static const char * const aszPre[] = {
        "mod_xyz.c", "mod_abc.c", NULL };

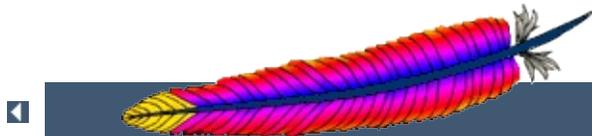
    ap_hook_do_something(my_something_doer, aszPre,
        NULL, APR_HOOK_MIDDLE);
}
```

Note that the sort used to achieve this is stable, so ordering set by

APR_HOOK_ORDER is preserved, as far as is possible.

Ben Laurie, 15th August 1999

| | | |



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Converting Modules from Apache 1.3 to Apache 2.0

This is a first attempt at writing the lessons I learned when trying to convert the `mod_mmap_static` module to Apache 2.0. It's by no means definitive and probably won't even be correct in some ways, but it's a start.



Cleanup Routines

These now need to be of type `apr_status_t` and return a value of that type. Normally the return value will be `APR_SUCCESS` unless there is some need to signal an error in the cleanup. Be aware that even though you signal an error not all code yet checks and acts upon the error.

Initialisation Routines

These should now be renamed to better signify where they sit in the overall process. So the name gets a small change from `mmap_init` to `mmap_post_config`. The arguments passed have undergone a radical change and now look like

- `apr_pool_t *p`
- `apr_pool_t *plog`
- `apr_pool_t *ptemp`
- `server_rec *s`

Data Types

A lot of the data types have been moved into the [APR](#). This means that some have had a name change, such as the one shown above. The following is a brief list of some of the changes that you are likely to have to make.

- `pool` becomes `apr_pool_t`
- `table` becomes `apr_table_t`



Register Hooks

The new architecture uses a series of hooks to provide for calling your functions. These you'll need to add to your module by way of a new function, `static void register_hooks(void)`. The function is really reasonably straightforward once you understand what needs to be done. Each function that needs calling at some stage in the processing of a request needs to be registered, handlers do not. There are a number of phases where functions can be added, and for each you can specify with a high degree of control the relative order that the function will be called in.

This is the code that was added to `mod_mmap_static`:

```
static void register_hooks(void)
{
    static const char * const aszPre[]={ "http_core.c'
    ap_hook_post_config(mmap_post_config,NULL,NULL,HOO
    ap_hook_translate_name(mmap_static_xlat,aszPre,NUI
};
```

This registers 2 functions that need to be called, one in the `post_config` stage (virtually every module will need this one) and one for the `translate_name` phase. note that while there are different function names the format of each is identical. So what is the format?

```
ap_hook_phase_name(function_name, predecessors,
successors, position);
```

There are 3 hook positions defined...

- `HOOK_FIRST`

- HOOK_MIDDLE
- HOOK_LAST

To define the position you use the position and then modify it with the predecessors and successors. Each of the modifiers can be a list of functions that should be called, either before the function is run (predecessors) or after the function has run (successors).

In the `mod_mmap_static` case I didn't care about the `post_config` stage, but the `mmap_static_xlat` **must** be called after the core module had done its name translation, hence the use of the `aszPre` to define a modifier to the position `HOOK_LAST`.

Module Definition

There are now a lot fewer stages to worry about when creating your module definition. The old definition looked like

```
module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD_MODULE_STUFF,
    /* initializer */
    /* dir config creator */
    /* dir merger --- default is to override */
    /* server config */
    /* merge server config */
    /* command handlers */
    /* handlers */
    /* filename translation */
    /* check_user_id */
    /* check auth */
    /* check access */
    /* type_checker */
    /* fixups */
    /* logger */
    /* header parser */
}
```

```

    /* child_init */
    /* child_exit */
    /* post read-request */
};

```

The new structure is a great deal simpler...

```

module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD20_MODULE_STUFF,
    /* create per-directory config structures */
    /* merge per-directory config structures */
    /* create per-server config structures */
    /* merge per-server config structures */
    /* command handlers */
    /* handlers */
    /* register hooks */
};

```

Some of these read directly across, some don't. I'll try to summarise what should be done below.

The stages that read directly across :

```

/* dir config creator */
    /* create per-directory config structures */
/* server config */
    /* create per-server config structures */
/* dir merger */
    /* merge per-directory config structures */
/* merge server config */
    /* merge per-server config structures */
/* command table */
    /* command apr_table_t */

```

```
/* handlers */  
    /* handlers */
```

The remainder of the old functions should be registered as hooks.
There are the following hook stages defined so far...

ap_hook_post_config

this is where the old `_init` routines get registered

ap_hook_http_method

retrieve the http method from a request. (legacy)

ap_hook_open_logs

open any specified logs

ap_hook_auth_checker

check if the resource requires authorization

ap_hook_access_checker

check for module-specific restrictions

ap_hook_check_user_id

check the user-id and password

ap_hook_default_port

retrieve the default port for the server

ap_hook_pre_connection

do any setup required just before processing, but after accepting

ap_hook_process_connection

run the correct protocol

ap_hook_child_init

call as soon as the child is started

ap_hook_create_request

??

ap_hook_fixups

last chance to modify things before generating content

ap_hook_handler

generate the content

ap_hook_header_parser

lets modules look at the headers, not used by most modules, because they use `post_read_request` for this

ap_hook_insert_filter

to insert filters into the filter chain

ap_hook_log_transaction

log information about the request

ap_hook_optional_fn_retrieve

retrieve any functions registered as optional

ap_hook_post_read_request

called after reading the request, before any other phase

ap_hook_quick_handler

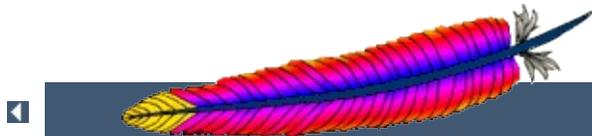
called before any request processing, used by cache modules.

ap_hook_translate_name

translate the URI into a filename

ap_hook_type_checker

determine and/or set the doc type



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

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 `/.///./` 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 OKDONE, the core will run the request filename against the `<Directory><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)) !=
        return decl_die(access_status, "check access",
    }

    if (ap_some_auth_required(r)) {
        if (((access_status = ap_run_check_user_id(r))
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type
                ? "check user. No user file
                : "perform authentication. /
                r);
        }

        if (((access_status = ap_run_auth_checker(r))
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type
                ? "check access. No groups
                : "perform authentication. /
                r);
        }
    }
    break;

case SATISFY_ANY:
    if (((access_status = ap_run_access_checker(r)) !=
        if (!ap_some_auth_required(r)) {
            return decl_die(access_status, "check acc
        }

        if (((access_status = ap_run_check_user_id(r))
            || !ap_auth_type(r)) {
```

```
        return decl_die(access_status, ap_auth_type(r)
                        ? "check user. No user file"
                        : "perform authentication. /"
                        r);
    }

    if (((access_status = ap_run_auth_checker(r))
        || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r)
                        ? "check access. No groups"
                        : "perform authentication. /"
                        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.



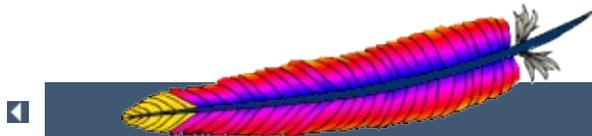
This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

Hook: insert_filter

Modules that transform the content in some way can insert their values and override existing filters, such that if the user configured a more advanced filter out-of-order, then the module can move its order as need be. There is no result code, so actions in this hook better be trusted to always succeed.

Hook: handler

The module finally has a chance to serve the request in its handler hook. Note that not every prepared request is sent to the handler hook. Many modules, such as [mod_autoindex](#), will create subrequests for a given URI, and then never serve the subrequest, but simply lists it for the user. Remember not to put required teardown from the hooks above into this module, but register pool cleanups against the request pool to free resources as required.



| | | |

◀ Apache HTTP Server 2.2

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

| | < > | ??? |

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.

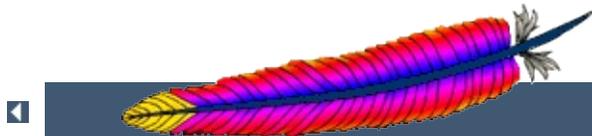
This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.



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.



Apache HTTP Server 2.2

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

| | | 200613 |



Apache



(Access Control)

Apache *URL* [_____](#)

(Algorithm)

(Cipher)

Apache(APache eXtension Tool) (apxs)

perl [\(module\)\(DSO\)](#)Apache web
[apxs](#)

Apache(Apache Portable Runtime) (APR)

APRApache HTTP Server
[Apache Portable Runtime Project](#)

(Authentication)

[_____](#)

(Certificate)

[X.509](#)([subject]) [\(Certification Authority\)](#)([issuer]) [_____](#)
[\(public key\)](#)(CA)CA
[SSL/TLS](#)

(Certificate Signing Request) (CSR)

[\(Certification Authority\)](#)CA[\(Private Key\)](#)(certificate)CSR
[SSL/TLS](#)

(Certification Authority) (CA)

CA
[SSL/TLS](#)

(Cipher)

DESIDEARC4
[SSL/TLS](#)

(Ciphertext)

[\(Plaintext\)](#)(Cipher)
[SSL/TLS](#)

(Common Gateway Interface) (CGI)

web [\(\)\(NCSA\)](#) [RFC](#)
[CGI](#)

(Configuration Directive)

[\(Directive\)](#)

(Configuration File)

Apache[\(Directives\)](#)

(CONNECT)

HTTPHTTP[\(method\)](#)SSL

(Context)

[\(Directives\)](#)

(Digital Signature)

[\(Certification Authority\)](#)*(Public Key)**(Certificate)* *(Private*
Key)*(CA)* CA
[SSL/TLS](#)

(Directive)

[\(Configuration File\)](#)Apache

(Dynamic Shared Object) (DSO)

Apache [httpd](#)*(Modules)*

(Environment Variable) (env-variable)

shellApacheApacheshell
[Apache](#)

(Export-Crippled)

(EAR)
[SSL/TLS](#)

(Filter)

application/octet-stream HTTPMIME Content -
Type([header](#))
[mod_mime](#)

(Module)

ApacheApache [httpd](#)(*static module*)(*dynamic module*)
[DSO](#)(*base module*)ApacheApache HTTP[tar](#)(*tarball*) (*third-*
party module)

(Module Magic Number) (MMN)

ApacheApacheAPIMMNApache

OpenSSL

SSL/TLS
<http://www.openssl.org/>

(Pass Phrase)

[\(Cipher\)](#)/
[SSL/TLS](#)

(Plaintext)

(Private Key)

[SSL/TLS](#)

(Proxy)

(*origin server*)
[mod_proxy](#)

(Public Key)

[SSL/TLS](#)

(Public Key Cryptography)

""(Asymmetric Cryptography)
[SSL/TLS](#)

(Regular Expression) (Regex)

"A""10" "Q"Apache

"images".gif .jpg "

/images/.*(jpg|gif)\$"A

[PCRE](#)Perl

(Reverse Proxy)

(*origin server*)([proxy](#))

(Secure Sockets Layer) (SSL)

NetscapeTCP/IP HTTPS SSL

[SSL/TLS](#)

(Server Side Includes) (SSI)

HTML

(Session)

SSLeay

Eric A. YoungSSL/TLS

(Symmetric Cryptography)

[SSL/TLS](#)

Tar(Tarball)

tarApachetarpkzip

(Transport Layer Security) (TLS)

Internet(IETF)SSLTCP/IPTLS1SSL3

[SSL/TLS](#)

(Uniform Resource Locator) (URL)

Internet/ ([Uniform Resource Identifier](#))URL http

https URL

http://httpd.apache.org/docs/2.2/glossary.html

(Uniform Resource Identifier) (URI)

[RFC 2396](#)URI [URL](#)

(Virtual Hosting)

Apache *IP(IP virtual hosting)IP* (*name-based virtual hosting*)
IP

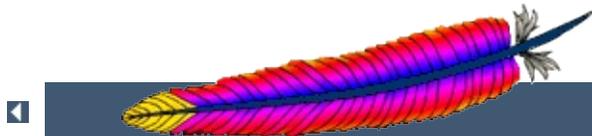
[Apache](#)

X.509

(ITU)SSL/TLS

[SSL/TLS](#)

| | | |



Apache HTTP Server 2.2

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

| | | 2006121 |

Apache

A | B | C | D | E | F | G | H | I | K | L | M | N | O | P |
R | S | T | U | V | W | X

- [AcceptFilter](#)
- [AcceptMutex](#)
- [AcceptPathInfo](#)
- [AccessFileName](#)
- [Action](#)
- [AddAlt](#)
- [AddAltByEncoding](#)
- [AddAltByType](#)
- [AddCharset](#)
- [AddDefaultCharset](#)
- [AddDescription](#)
- [AddEncoding](#)
- [AddHandler](#)
- [AddIcon](#)
- [AddIconByEncoding](#)
- [AddIconByType](#)
- [AddInputFilter](#)
- [AddLanguage](#)
- [AddModuleInfo](#)
- [AddOutputFilter](#)
- [AddOutputFilterByType](#)
- [AddType](#)
- [Alias](#)
- [AliasMatch](#)
- [Allow](#)

- [AllowCONNECT](#)
- [AllowEncodedSlashes](#)
- [AllowOverride](#)
- [Anonymous](#)
- [Anonymous_LogEmail](#)
- [Anonymous_MustGiveEmail](#)
- [Anonymous_NoUserID](#)
- [Anonymous_VerifyEmail](#)
- [AuthBasicAuthoritative](#)
- [AuthBasicProvider](#)
- [AuthDBDUserPWQuery](#)
- [AuthDBDUserRealmQuery](#)
- [AuthDBMGroupFile](#)
- [AuthDBMType](#)
- [AuthDBMUserFile](#)
- [AuthDefaultAuthoritative](#)
- [AuthDigestAlgorithm](#)
- [AuthDigestDomain](#)
- [AuthDigestNcCheck](#)
- [AuthDigestNonceFormat](#)
- [AuthDigestNonceLifetime](#)
- [AuthDigestProvider](#)
- [AuthDigestQop](#)
- [AuthDigestShmemSize](#)
- [AuthGroupFile](#)
- [AuthLDAPBindDN](#)
- [AuthLDAPBindPassword](#)
- [AuthLDAPCharsetConfig](#)
- [AuthLDAPCompareDNOnServer](#)
- [AuthLDAPDereferenceAliases](#)
- [AuthLDAPGroupAttribute](#)
- [AuthLDAPGroupAttributeIsDN](#)
- [AuthLDAPRemoteUserIsDN](#)
- [AuthLDAPUrl](#)

- [AuthName](#)
- [<AuthnProviderAlias>](#)
- [AuthType](#)
- [AuthUserFile](#)
- [AuthzDBMAuthoritative](#)
- [AuthzDBMType](#)
- [AuthzDefaultAuthoritative](#)
- [AuthzGroupFileAuthoritative](#)
- [AuthzLDAPAuthoritative](#)
- [AuthzOwnerAuthoritative](#)
- [AuthzUserAuthoritative](#)
- [BrowserMatch](#)
- [BrowserMatchNoCase](#)
- [BufferedLogs](#)
- [CacheDefaultExpire](#)
- [CacheDirLength](#)
- [CacheDirLevels](#)
- [CacheDisable](#)
- [CacheEnable](#)
- [CacheFile](#)
- [CacheIgnoreCacheControl](#)
- [CacheIgnoreHeaders](#)
- [CacheIgnoreNoLastMod](#)
- [CacheLastModifiedFactor](#)
- [CacheMaxExpire](#)
- [CacheMaxFileSize](#)
- [CacheMinFileSize](#)
- [CacheNegotiatedDocs](#)
- [CacheRoot](#)
- [CacheStoreNoStore](#)
- [CacheStorePrivate](#)
- [CGIMapExtension](#)
- [CharsetDefault](#)
- [CharsetOptions](#)

- [CharsetSourceEnc](#)
- [CheckSpelling](#)
- [ContentDigest](#)
- [CookieDomain](#)
- [CookieExpires](#)
- [CookieLog](#)
- [CookieName](#)
- [CookieStyle](#)
- [CookieTracking](#)
- [CoreDumpDirectory](#)
- [CustomLog](#)
- [Dav](#)
- [DavDepthInfinity](#)
- [DavGenericLockDB](#)
- [DavLockDB](#)
- [DavMinTimeout](#)
- [DBDExptime](#)
- [DBDKeep](#)
- [DBDMax](#)
- [DBDMin](#)
- [DBDParams](#)
- [DBDPersist](#)
- [DBDPrepareSQL](#)
- [DBDriver](#)
- [DefaultIcon](#)
- [DefaultLanguage](#)
- [DefaultType](#)
- [DeflateBufferSize](#)
- [DeflateCompressionLevel](#)
- [DeflateFilterNote](#)
- [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>](#)
- [FilterChain](#)
- [FilterDeclare](#)
- [FilterProtocol](#)
- [FilterProvider](#)
- [FilterTrace](#)
- [ForceLanguagePriority](#)
- [ForceType](#)
- [ForensicLog](#)
- [GracefulShutdownTimeout](#)
- [Group](#)
- [Header](#)
- [HeaderName](#)
- [HostnameLookups](#)

- [IdentityCheck](#)
- [IdentityCheckTimeout](#)
- [<IfDefine>](#)
- [<IfModule>](#)
- [<IfVersion>](#)
- [ImapBase](#)
- [ImapDefault](#)
- [ImapMenu](#)
- [Include](#)
- [IndexIgnore](#)
- [IndexOptions](#)
- [IndexOrderDefault](#)
- [IndexStyleSheet](#)
- [ISAPIAppendLogToErrors](#)
- [ISAPIAppendLogToQuery](#)
- [ISAPICacheFile](#)
- [ISAPIFakeAsync](#)
- [ISAPILogNotSupported](#)
- [ISAPIReadAheadBuffer](#)
- [KeepAlive](#)
- [KeepAliveTimeout](#)
- [LanguagePriority](#)
- [LDAPCacheEntries](#)
- [LDAPCacheTTL](#)
- [LDAPConnectionTimeout](#)
- [LDAPOpCacheEntries](#)
- [LDAPOpCacheTTL](#)
- [LDAPSharedCacheFile](#)
- [LDAPSharedCacheSize](#)
- [LDAPTrustedClientCert](#)
- [LDAPTrustedGlobalCert](#)
- [LDAPTrustedMode](#)
- [LDAPVerifyServerCert](#)
- [<Limit>](#)

- [<LimitExcept>](#)
- [LimitInternalRecursion](#)
- [LimitRequestBody](#)
- [LimitRequestFields](#)
- [LimitRequestFieldSize](#)
- [LimitRequestLine](#)
- [LimitXMLRequestBody](#)
- [Listen](#)
- [ListenBackLog](#)
- [LoadFile](#)
- [LoadModule](#)
- [<Location>](#)
- [<LocationMatch>](#)
- [LockFile](#)
- [LogFormat](#)
- [LogLevel](#)
- [MaxClients](#)
- [MaxKeepAliveRequests](#)
- [MaxMemFree](#)
- [MaxRequestsPerChild](#)
- [MaxRequestsPerThread](#)
- [MaxSpareServers](#)
- [MaxSpareThreads](#)
- [MaxThreads](#)
- [MCacheMaxObjectCount](#)
- [MCacheMaxObjectSize](#)
- [MCacheMaxStreamingBuffer](#)
- [MCacheMinObjectSize](#)
- [MCacheRemovalAlgorithm](#)
- [MCacheSize](#)
- [MetaDir](#)
- [MetaFiles](#)
- [MetaSuffix](#)
- [MimeMagicFile](#)

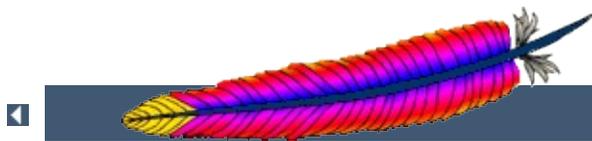
- [MinSpareServers](#)
- [MinSpareThreads](#)
- [MMapFile](#)
- [ModMimeUsePathInfo](#)
- [MultiviewsMatch](#)
- [NameVirtualHost](#)
- [NoProxy](#)
- [NWSSLTrustedCerts](#)
- [NWSSLUpgradeable](#)
- [Options](#)
- [Order](#)
- [PassEnv](#)
- [PidFile](#)
- [ProtocolEcho](#)
- [<Proxy>](#)
- [ProxyBadHeader](#)
- [ProxyBlock](#)
- [ProxyDomain](#)
- [ProxyErrorOverride](#)
- [ProxyIOBufferSize](#)
- [<ProxyMatch>](#)
- [ProxyMaxForwards](#)
- [ProxyPass](#)
- [ProxyPassReverse](#)
- [ProxyPassReverseCookieDomain](#)
- [ProxyPassReverseCookiePath](#)
- [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](#)
- [SSLCADNRequestFile](#)
- [SSLCADNRequestPath](#)
- [SSLCARevocationFile](#)
- [SSLCARevocationPath](#)
- [SSLCertificateChainFile](#)
- [SSLCertificateFile](#)
- [SSLCertificateKeyFile](#)
- [SSLCipherSuite](#)
- [SSLCryptoDevice](#)

- [SSLEngine](#)
- [SSLHonorCipherOrder](#)
- [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](#)
- [UseCanonicalPhysicalPort](#)
- [User](#)
- [UserDir](#)
- [VirtualDocumentRoot](#)
- [VirtualDocumentRootIP](#)
- [<VirtualHost>](#)
- [VirtualScriptAlias](#)
- [VirtualScriptAliasIP](#)
- [Win32DisableAcceptEx](#)
- [XBitHack](#)



| | | |



Apache HTTP Server 2.2

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

| | | ??? |



()"+"

<table border="1"> <tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td><td>I</td></tr> <tr><td>K</td><td>L</td><td>M</td><td>N</td><td>O</td><td>P</td><td>R</td><td>S</td><td>T</td></tr> <tr><td>U</td><td>V</td><td>W</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	R	S	T	U	V	W	X						s	server config	C	
	A	B	C	D	E	F	G	H	I																						
	K	L	M	N	O	P	R	S	T																						
	U	V	W	X																											
v	virtual host	M	MPM																												
d	directory	B																													
h	.htaccess	E																													
			X																												

AcceptFilter <i>protocol accept filter</i>	
Socket	
AcceptMutex <i>Default method</i>	Default
Apache()(socket)	
AcceptPathInfo <i>On Off Default</i>	Default
AccessFileName <i>filename [filename] ...</i>	.htaccess
Action <i>action-type cgi-script [virtual]</i>	
CGI	
AddAlt <i>string file [file] ...</i>	
Alternate text to display for a file, instead of an icon selected by file	
AddAltByEncoding <i>string MIME-encoding [MIME-encoding] ...</i>	
Alternate text to display for a file instead of an icon selected by MIM	
AddAltByType <i>string MIME-type [MIME-type]</i>	
...	
Alternate text to display for a file, instead of an icon selected by MIM	

type	
AddCharset <i>charset extension [extension] ...</i>	
AddDefaultCharset On Off <i>charset</i> text/plain/text/html/HTTP	Off
AddDescription <i>string file [file] ...</i> Description to display for a file	
AddEncoding <i>MIME-enc extension [extension]</i> ...	
AddHandler <i>handler-name extension [extension] ...</i>	
AddIcon <i>icon name [name] ...</i> Icon to display for a file selected by name	
AddIconByEncoding <i>icon MIME-encoding [MIME-encoding] ...</i> Icon to display next to files selected by MIME content-encoding	
AddIconByType <i>icon MIME-type [MIME-type]</i> ...	
AddInputFilter <i>filter[:filter...] extension [extension] ...</i>	
AddLanguage <i>MIME-lang extension [extension] ...</i>	
AddModuleInfo <i>module-name string</i> server-info	
AddOutputFilter <i>filter[:filter...] extension [extension] ...</i>	

AddOutputFilterByType filter[:filter...] MIME-type [MIME-type] ...	
MIME	
AddType MIME-type extension [extension] ...	
Alias URL-path file-path directory-path	
URL	
AliasMatch regex file-path directory-path	
URL	
Allow from all host env=env-variable [host env=env-variable] ...	
AllowCONNECT port [port] ...	443 563
CONNECT	
AllowEncodedSlashes On Off	Off
URL	
AllowOverride All None directive-type [directive-type] ...	All
.htaccess	
Anonymous user [user] ...	
Specifies userIDs that are allowed access without password verification	
Anonymous_LogEmail On Off	On
Sets whether the password entered will be logged in the error log	
Anonymous_MustGiveEmail On Off	On
Specifies whether blank passwords are allowed	
Anonymous_NoUserID On Off	Off
Sets whether the userID field may be empty	
Anonymous_VerifyEmail On Off	Off
Sets whether to check the password field for a correctly formatted email address	

AuthBasicAuthoritative On Off ()	On
AuthBasicProvider <i>provider-name</i> [<i>provider-name</i>] ... ()(Provider)	file
AuthDBDUserPWQuery <i>query</i> SQL query to look up a password for a user	
AuthDBDUserRealmQuery <i>query</i> SQL query to look up a password hash for a user and realm.	
AuthDBMGroupFile <i>file-path</i> Sets the name of the database file containing the list of user groups authorization	
AuthDBMType default SDBM GDBM NDBM DB Sets the type of database file that is used to store passwords	default
AuthDBMUserFile <i>file-path</i> Sets the name of a database file containing the list of users and pas authentication	
AuthDefaultAuthoritative On Off	On
AuthDigestAlgorithm MD5 MD5-sess	MD5
AuthDigestDomain <i>URI</i> [<i>URI</i>] ... URI	
AuthDigestNcCheck On Off Enables or disables checking of the nonce-count sent by the server	Off
AuthDigestNonceFormat <i>format</i> Determines how the nonce is generated	
AuthDigestNonceLifetime <i>seconds</i> nonce()	300
AuthDigestProvider <i>provider-name</i> [<i>provider-</i>	file

name] ...	
(Provider)	
AuthDigestQop none auth auth-int [auth auth-int]	auth
AuthDigestShmemSize size	1000
AuthGroupFile file-path	
AuthLDAPBindDN distinguished-name	
Optional DN to use in binding to the LDAP server	
AuthLDAPBindPassword password	
Password used in conjunction with the bind DN	
AuthLDAPCharsetConfig file-path	
Language to charset conversion configuration file	
AuthLDAPCompareDNOnServer on off	on
Use the LDAP server to compare the DN's	
AuthLDAPDereferenceAliases never searching finding always	Always
When will the module de-reference aliases	
AuthLDAPGroupAttribute attribute	
LDAP attributes used to check for group membership	
AuthLDAPGroupAttributeIsDN on off	on
Use the DN of the client username when checking for group membership	
AuthLDAPRemoteUserIsDN on off	off
Use the DN of the client username to set the REMOTE_USER environment variable	
AuthLDAPUrl url [NONE SSL TLS STARTTLS]	
URL specifying the LDAP search parameters	
AuthName auth-domain	
HTTP	

<AuthnProviderAlias <i>baseProvider Alias</i>> ...	
</AuthnProviderAlias>	
AuthType Basic Digest	
AuthUserFile <i>file-path</i>	
/	
AuthzDBMAuthoritative On Off	On
Sets whether authorization will be passed on to lower level modules	
AuthzDBMType	default
default SDBM GDBM NDBM DB	
Sets the type of database file that is used to store list of user groups	
AuthzDefaultAuthoritative On Off	On
AuthzGroupFileAuthoritative On Off	On
AuthzLDAPAuthoritative on off	on
Prevent other authentication modules from authenticating the user if fails	
AuthzOwnerAuthoritative On Off	On
AuthzUserAuthoritative On Off	On
BrowserMatch <i>regex [!]</i>env-variable[=<i>value</i>] [!] env-variable[=<i>value</i>] ...	
User-Agent	
BrowserMatchNoCase <i>regex [!]</i>env-variable[=<i>value</i>] [!] env-variable[=<i>value</i>] ...	
User-Agent	
BufferedLogs On Off	Off

<u>CacheDefaultExpire</u> <i>seconds</i>	3600 (one hour)
The default duration to cache a document when no expiry date is specified	
<u>CacheDirLength</u> <i>length</i>	2
The number of characters in subdirectory names	
<u>CacheDirLevels</u> <i>levels</i>	3
The number of levels of subdirectories in the cache.	
<u>CacheDisable</u> <i>url-string</i>	
Disable caching of specified URLs	
<u>CacheEnable</u> <i>cache_type url-string</i>	
Enable caching of specified URLs using a specified storage manager	
<u>CacheFile</u> <i>file-path [file-path] ...</i>	
Cache a list of file handles at startup time	
<u>CacheIgnoreCacheControl</u> On Off	Off
Ignore request to not serve cached content to client	
<u>CacheIgnoreHeaders</u> <i>header-string [header-string] ...</i>	None
Do not store the given HTTP header(s) in the cache.	
<u>CacheIgnoreNoLastMod</u> On Off	Off
Ignore the fact that a response has no Last Modified header.	
<u>CacheLastModifiedFactor</u> <i>float</i>	0.1
The factor used to compute an expiry date based on the LastModified header	
<u>CacheMaxExpire</u> <i>seconds</i>	86400 (one day)
The maximum time in seconds to cache a document	
<u>CacheMaxFileSize</u> <i>bytes</i>	1000000
The maximum size (in bytes) of a document to be placed in the cache	
<u>CacheMinFileSize</u> <i>bytes</i>	1
The minimum size (in bytes) of a document to be placed in the cache	
<u>CacheNegotiatedDocs</u> On Off	Off
<u>CacheRoot</u> <i>directory</i>	
The directory root under which cache files are stored	

CacheStoreNoStore On Off	Off
Attempt to cache requests or responses that have been marked as private	
CacheStorePrivate On Off	Off
Attempt to cache responses that the server has marked as private	
CGIMapExtension <i>cgi-path .extension</i>	
CGI	
CharsetDefault <i>charset</i>	
Charset to translate into	
CharsetOptions <i>option [option] ...</i>	DebugLevel=0 NoImp
Configures charset translation behavior	
CharsetSourceEnc <i>charset</i>	
Source charset of files	
CheckSpelling on off	Off
Enables the spelling module	
ContentDigest On Off	Off
Content-MD5	
CookieDomain <i>domain</i>	
The domain to which the tracking cookie applies	
CookieExpires <i>expiry-period</i>	
Expiry time for the tracking cookie	
CookieLog <i>filename</i>	
cookies	
CookieName <i>token</i>	Apache
Name of the tracking cookie	
CookieStyle	Netscape
Netscape Cookie Cookie2 RFC2109 RFC2965	
Format of the cookie header field	
CookieTracking on off	off
Enables tracking cookie	
CoreDumpDirectory <i>directory</i>	
Apache	

CustomLog <i>file pipe format nickname</i> [env= [!]<i>environment-variable</i>]	
Dav On Off <i>provider-name</i> Enable WebDAV HTTP methods	Off
DavDepthInfinity on off Allow PROPFIND, Depth: Infinity requests	off
DavGenericLockDB <i>file-path</i> Location of the DAV lock database	
DavLockDB <i>file-path</i> Location of the DAV lock database	
DavMinTimeout <i>seconds</i> Minimum amount of time the server holds a lock on a DAV resource	0
DBDExptime <i>time-in-seconds</i> Keepalive time for idle connections	
DBDKeep <i>number</i> Maximum sustained number of connections	
DBDMax <i>number</i> Maximum number of connections	
DBDMin <i>number</i> Minimum number of connections	
DBDParams <i>param1=value1[,param2=value2]</i> Parameters for database connection	
DBDPersist 0 1 Whether to use persistent connections	
DBDPrepareSQL "<i>SQL statement</i>" <i>label</i> Define an SQL prepared statement	
DBDriver <i>name</i> Specify an SQL driver	
DefaultIcon <i>url-path</i> Icon to display for files when no specific icon is configured	

DefaultLanguage <i>MIME-lang</i>	
DefaultType <i>MIME-type</i> MIME	text/plain
DeflateBufferSize <i>value</i> zlib()	8096
DeflateCompressionLevel <i>value</i>	
DeflateFilterNote [<i>type</i>] <i>notename</i>	
DeflateMemLevel <i>value</i> zlib	9
DeflateWindowSize <i>value</i> Zlib(compression window)	15
Deny from all <i>host</i> <i>env=env-variable</i> <i>[host env=env-variable]</i> ...	
<Directory <i>directory-path</i> ... </Directory>	
DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...	index.html
<DirectoryMatch <i>regex</i> ... </DirectoryMatch>	
DirectorySlash On Off (/)	On
DocumentRoot <i>directory-path</i>	/usr/local/apache/h +
DumpIOInput On Off	Off
DumpIOOutput On Off	Off

EnableExceptionHook On Off	Off
EnableMMAP On Off (memory-mapping)	On
EnableSendfile On Off sendfile	On
ErrorDocument <i>error-code</i> <i>document</i>	
ErrorLog <i>file-path</i> syslog[:<i>facility</i>]	logs/error_log (Uni +
Example Demonstration directive to illustrate the Apache module API	
ExpiresActive On Off "Expires:""Cache-Control:"	
ExpiresByType <i>MIME-type</i> <code><code></code><i>seconds</i> MIMEExpires	
ExpiresDefault <code><code></code><i>seconds</i>	
ExtendedStatus On Off Keep track of extended status information for each request	Off
ExtFilterDefine <i>filtername</i> <i>parameters</i> Define an external filter	
ExtFilterOptions <i>option</i> [<i>option</i>] ... Configure <code>mod_ext_filter</code> options	DebugLevel=0 NoLog +
FileETag <i>component</i> ... ETag	INode MTime Size
<Files <i>filename</i>> ... </Files>	
<FilesMatch <i>regex</i>> ... </FilesMatch>	

FilterChain <i>[+@-@!]filter-name ...</i>	
Configure the filter chain	
FilterDeclare <i>filter-name [type]</i>	
Declare a smart filter	
FilterProtocol <i>filter-name [provider-name] proto-flags</i>	
Deal with correct HTTP protocol handling	
FilterProvider <i>filter-name provider-name [req resp env]=dispatch match</i>	
Register a content filter	
FilterTrace <i>filter-name level</i>	
Get debug/diagnostic information from <code>mod filter</code>	
ForceLanguagePriority <i>None Prefer Fallback [Prefer Fallback]</i>	Prefer
ForceType <i>MIME-type None</i>	
MIME	
ForensicLog <i>filename pipe</i>	
Sets filename of the forensic log	
GracefulShutdownTimeout <i>seconds</i>	
Group <i>unix-group</i>	#-1
Apache	
Header <i>[condition] set append add unset echo header [value] [early env=[!]variable]</i>	
HTTP	
HeaderName <i>filename</i>	
Name of the file that will be inserted at the top of the index listing	
HostnameLookups <i>On Off Double</i>	Off
IPDNS	
IdentityCheck <i>On Off</i>	Off

RFC1413	
IdentityCheckTimeout <i>seconds</i>	30
Determines the timeout duration for ident requests	
<IfDefine [!]parameter-name> ... </IfDefine>	
<IfModule [!]module-file module-identifier> ... </IfModule>	
<IfVersion [!]operator version> ... </IfVersion>	
contains version dependent configuration	
ImapBase <i>map referer URL</i>	http://servername/
Default base for imagemap files	
ImapDefault <i>error nocontent map referer URL</i>	nocontent
Default action when an imagemap is called with coordinates that are explicitly mapped	
ImapMenu <i>none formatted semiformatted unformatted</i>	
Action if no coordinates are given when calling an imagemap	
Include <i>file-path directory-path</i>	
IndexIgnore <i>file [file] ...</i>	
Adds to the list of files to hide when listing a directory	
IndexOptions <i>[+ -]option [[+ -]option] ...</i>	
Various configuration settings for directory indexing	
IndexOrderDefault <i>Ascending Descending</i> <i>Name Date Size Description</i>	Ascending Name
Sets the default ordering of the directory index	
IndexStyleSheet <i>url-path</i>	
Adds a CSS stylesheet to the directory index	
ISAPIAppendLogToErrors <i>on off</i>	off

ISAPIHSE_APPEND_LOG_PARAMETER	
ISAPIAppendLogToQuery on off	on
ISAPIHSE_APPEND_LOG_PARAMETER	
ISAPICacheFile file-path [file-path] ...	
ISAPI	
ISAPIFakeAsync on off	off
ISAPI	
ISAPILogNotSupported on off	off
ISAPI	
ISAPIReadAheadBuffer size	49152
ISAPI	
KeepAlive On Off	On
HTTP	
KeepAliveTimeout seconds	5
LanguagePriority MIME-lang [MIME-lang] ...	
LDAPCacheEntries number	1024
LDAP	
LDAPCacheTTL seconds	600
search/bind	
LDAPConnectionTimeout seconds	
LDAPOpCacheEntries number	1024
LDAP compare	
LDAPOpCacheTTL seconds	600
LDAPSharedCacheFile directory-path/filename	
LDAPSharedCacheSize bytes	102400

<u>LDAPTrustedClientCert</u> <i>type</i> <u>directory-path/filename/nickname</u> <u>[password]</u>	
Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates	
<u>LDAPTrustedGlobalCert</u> <i>type</i> <u>directory-path/filename</u> <u>[password]</u>	
Sets the file or database containing global trusted Certificate Authority client certificates	
<u>LDAPTrustedMode</u> <i>type</i>	
Specifies the SSL/TLS mode to be used when connecting to an LDAP server	
<u>LDAPVerifyServerCert</u> <i>On Off</i>	On
Force server certificate verification	
<u><Limit method [method] ... > ... </Limit></u>	
HTTP	
<u><LimitExcept method [method] ... > ... </LimitExcept></u>	
HTTP	
<u>LimitInternalRecursion</u> <i>number</i> <u>[number]</u>	10
<u>LimitRequestBody</u> <i>bytes</i>	0
HTTP	
<u>LimitRequestFields</u> <i>number</i>	100
HTTP	
<u>LimitRequestFieldsize</u> <i>bytes</i>	
<u>LimitRequestLine</u> <i>bytes</i>	8190
HTTP	
<u>LimitXMLRequestBody</u> <i>bytes</i>	1000000
XML	
<u>Listen</u> <u>[IP-address:]portnumber</u> <u>[protocol]</u>	

IP	
ListenBacklog <i>backlog</i> (pending connection)	
LoadFile <i>filename [filename] ...</i>	
LoadModule <i>module filename</i>	
<Location <i>URL-path URL</i>> ... </Location> URL	
<LocationMatch <i>regex</i>> ... </LocationMatch> URL	
LockFile <i>filename</i>	logs/accept.lock
LogFormat <i>format nickname [nickname]</i>	"%h %l %u %t \"%r\" +
LogLevel <i>level</i>	warn
MaxClients <i>number</i>	
MaxKeepAliveRequests <i>number</i>	100
MaxMemFree <i>KBytes</i> free()(KB)	0
MaxRequestsPerChild <i>number</i>	10000
MaxRequestsPerThread <i>number</i> Limit on the number of requests that an individual thread will handle life	0
MaxSpareServers <i>number</i>	10
MaxSpareThreads <i>number</i>	

<u>MaxThreads</u> <i>number</i>	2048
Set the maximum number of worker threads	
<u>MCacheMaxObjectCount</u> <i>value</i>	1009
<u>MCacheMaxObjectSize</u> <i>bytes</i>	10000
()	
<u>MCacheMaxStreamingBuffer</u> <i>size in bytes</i>	the smaller of 1000 +
<u>MCacheMinObjectSize</u> <i>bytes</i>	0
()	
<u>MCacheRemovalAlgorithm</u> LRU GDSF	GDSF
<u>MCacheSize</u> <i>KBytes</i>	100
KB	
<u>MetaDir</u> <i>directory</i>	.web
Name of the directory to find CERN-style meta information files	
<u>MetaFiles</u> on off	off
Activates CERN meta-file processing	
<u>MetaSuffix</u> <i>suffix</i>	.meta
File name suffix for the file containing CERN-style meta information	
<u>MimeMagicFile</u> <i>file-path</i>	
MagicMIME	
<u>MinSpareServers</u> <i>number</i>	5
<u>MinSpareThreads</u> <i>number</i>	
<u>MMapFile</u> <i>file-path [file-path] ...</i>	
Map a list of files into memory at startup time	
<u>ModMimeUsePathInfo</u> On Off	Off
path_info	

MultiviewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters] MultiViews	NegotiatedOnly
NameVirtualHost <i>addr[:port]</i> IP()	
NoProxy <i>host [host] ...</i> //	
NWSSLTrustedCerts <i>filename [filename] ...</i>	
NWSSLUpgradeable [<i>IP-address:</i>] <i>portnumber</i> SSL	
Options [<i>+ -option</i> [<i>[+ -option]</i> ...	All
Order <i>ordering</i> AllowDeny	Deny,Allow
PassEnv <i>env-variable [env-variable] ...</i> shell	
PidFile <i>filename</i> (<i>)</i> PID	logs/httpd.pid
ProtocolEcho On Off Turn the echo server on or off	
<Proxy <i>wildcard-url</i></Proxy>	
ProxyBadHeader IsError Ignore StartBody	IsError
ProxyBlock * <i>word host domain</i> [word host domain] ...	
ProxyDomain <i>Domain</i>	

ProxyErrorOverride <i>On Off</i>	Off
ProxyIOBufferSize <i>bytes</i>	8192
<ProxyMatch regex> ...</ProxyMatch>	
ProxyMaxForwards <i>number</i>	10
ProxyPass [<i>path</i>] <i>! url</i> [<i>key=value key=value ...</i>] URL	
ProxyPassReverse [<i>path</i>] <i>url</i> HTTPURL	
ProxyPassReverseCookieDomain <i>internal-domain public-domain</i> Adjusts the Domain string in Set-Cookie headers from a reverse- proxy server	
ProxyPassReverseCookiePath <i>internal-path public-path</i> Adjusts the Path string in Set-Cookie headers from a reverse- proxy server	
ProxyPreserveHost <i>On Off</i> HTTP	Off
ProxyReceiveBufferSize <i>bytes</i> HTTPFTP()	0
ProxyRemote <i>match remote-server</i>	
ProxyRemoteMatch <i>regex remote-server</i>	
ProxyRequests <i>On Off</i> ()	Off
ProxyTimeout <i>seconds</i>	300

ProxyVia <i>On Off Full Block</i>	Off
Via	
ReadmeName <i>filename</i>	
Name of the file that will be inserted at the end of the index listing	
ReceiveBufferSize <i>bytes</i>	0
TCP()	
Redirect [<i>status</i>] <i>URL-path URL</i>	
URL	
RedirectMatch [<i>status</i>] <i>regex URL</i>	
URL	
RedirectPermanent <i>URL-path URL</i>	
URL	
RedirectTemp <i>URL-path URL</i>	
URL	
RemoveCharset <i>extension [extension] ...</i>	
RemoveEncoding <i>extension [extension] ...</i>	
RemoveHandler <i>extension [extension] ...</i>	
RemoveInputFilter <i>extension [extension] ...</i>	
RemoveLanguage <i>extension [extension] ...</i>	
RemoveOutputFilter <i>extension [extension] ...</i>	
RemoveType <i>extension [extension] ...</i>	
RequestHeader <i>set append add unset header [value] [early env=[!]variable]</i>	

HTTP	
Require <i>entity-name</i> [<i>entity-name</i>] ...	
RewriteBase <i>URL-path</i>	
Sets the base URL for per-directory rewrites	
RewriteCond <i>TestString CondPattern</i>	
Defines a condition under which rewriting will take place	
RewriteEngine on off	off
Enables or disables runtime rewriting engine	
RewriteLock <i>file-path</i>	
Sets the name of the lock file used for RewriteMap synchronization	
RewriteLog <i>file-path</i>	
Sets the name of the file used for logging rewrite engine processing	
RewriteLogLevel <i>Level</i>	0
Sets the verbosity of the log file used by the rewrite engine	
RewriteMap <i>MapName MapType:MapSource</i>	
Defines a mapping function for key-lookup	
RewriteOptions <i>Options</i>	
Sets some special options for the rewrite engine	
RewriteRule <i>Pattern Substitution</i>	
Defines rules for the rewriting engine	
RLimitCPU <i>seconds max</i> [<i>seconds max</i>]	
ApacheCPU	
RLimitMEM <i>bytes max</i> [<i>bytes max</i>]	
Apache	
RLimitNPROC <i>number max</i> [<i>number max</i>]	
Apache	
Satisfy Any All	All
ScoreBoardFile <i>file-path</i>	logs/apache_status
(coordination data)	

Script method <i>cgi-script</i> CGI	
ScriptAlias <i>URL-path file-path directory-path</i> URLCGI	
ScriptAliasMatch <i>regex file-path directory-path</i> URLCGI	
ScriptInterpreterSource <i>Registry Registry-Strict Script</i> CGI	Script
ScriptLog <i>file-path</i> CGI	
ScriptLogBuffer <i>bytes</i> PUTPOST	1024
ScriptLogLength <i>bytes</i> ()	10385760
ScriptSock <i>file-path</i> CGI	logs/cgisock
SecureListen [<i>IP-address:</i>]<i>portnumber</i> Certificate-Name [MUTUAL] SSL	
SendBufferSize <i>bytes</i> TCP()	0
ServerAdmin <i>email-address URL</i>	
ServerAlias <i>hostname [hostname] ...</i>	
ServerLimit <i>number</i>	
ServerName <i>fully-qualified-domain-name[:port]</i>	

ServerPath <i>URL-path</i> URL	
ServerRoot <i>directory-path</i>	/usr/local/apache
ServerSignature On Off EMail	Off
ServerTokens Major Minor Min[imal] Prod[uctOnly] OS Full "Server:"	Full
SetEnv <i>env-variable value</i>	
SetEnvIf <i>attribute regex [!]env-variable[=value] [!]env-variable[=value]] ...</i>	
SetEnvIfNoCase <i>attribute regex [!]env-variable[=value] [!]env-variable[=value]] ...</i>	
SetHandler <i>handler-name None</i>	
SetInputFilter <i>filter[:filter...]</i> POST	
SetOutputFilter <i>filter[:filter...]</i>	
SSIEndTag <i>tag</i> String that ends an include element	"-->"
SSSErrorMsg <i>message</i> Error message displayed when there is an SSI error	"[an error occurred +
SSIStartTag <i>tag</i> String that starts an include element	"<!--#"
SSITimeFormat <i>formatstring</i>	"%A, %d-%b-%Y %H:%M +

Configures the format in which date strings are displayed	
<u>SSIUndefinedEcho</u> <i>string</i>	"(none)"
String displayed when an unset variable is echoed	
<u>SSLCACertificateFile</u> <i>file-path</i>	
File of concatenated PEM-encoded CA Certificates for Client Auth	
<u>SSLCACertificatePath</u> <i>directory-path</i>	
Directory of PEM-encoded CA Certificates for Client Auth	
<u>SSLCADNRequestFile</u> <i>file-path</i>	
File of concatenated PEM-encoded CA Certificates for defining acceptable C	
<u>SSLCADNRequestPath</u> <i>directory-path</i>	
Directory of PEM-encoded CA Certificates for defining acceptable C	
<u>SSLCARevocationFile</u> <i>file-path</i>	
File of concatenated PEM-encoded CA CRLs for Client Auth	
<u>SSLCARevocationPath</u> <i>directory-path</i>	
Directory of PEM-encoded CA CRLs for Client Auth	
<u>SSLCertificateChainFile</u> <i>file-path</i>	
File of PEM-encoded Server CA Certificates	
<u>SSLCertificateFile</u> <i>file-path</i>	
Server PEM-encoded X.509 Certificate file	
<u>SSLCertificateKeyFile</u> <i>file-path</i>	
Server PEM-encoded Private Key file	
<u>SSLCipherSuite</u> <i>cipher-spec</i>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL handshake	
<u>SSLCryptoDevice</u> <i>engine</i>	builtin
Enable use of a cryptographic hardware accelerator	
<u>SSLEngine</u> <i>on off optional</i>	off
SSL Engine Operation Switch	
<u>SSLHonorCipherOrder</u> <i>flag</i>	
Option to prefer the server's cipher preference order	

<u>SSLMutex type</u>	none
Semaphore for internal mutual exclusion of operations	
<u>SSLOptions [+]-option ...</u>	
Configure various SSL engine run-time options	
<u>SSLPassPhraseDialog type</u>	builtin
Type of pass phrase dialog for encrypted private keys	
<u>SSLProtocol [+]-protocol ...</u>	all
Configure usable SSL protocol flavors	
<u>SSLProxyCACertificateFile file-path</u>	
File of concatenated PEM-encoded CA Certificates for Remote Serv	
<u>SSLProxyCACertificatePath directory-path</u>	
Directory of PEM-encoded CA Certificates for Remote Server Auth	
<u>SSLProxyCARevocationFile file-path</u>	
File of concatenated PEM-encoded CA CRLs for Remote Server Au	
<u>SSLProxyCARevocationPath directory-path</u>	
Directory of PEM-encoded CA CRLs for Remote Server Auth	
<u>SSLProxyCipherSuite cipher-spec</u>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL proxy handshake	
<u>SSLProxyEngine on off</u>	off
SSL Proxy Engine Operation Switch	
<u>SSLProxyMachineCertificateFile filename</u>	
File of concatenated PEM-encoded client certificates and keys to be the proxy	
<u>SSLProxyMachineCertificatePath directory</u>	
Directory of PEM-encoded client certificates and keys to be used by	
<u>SSLProxyProtocol [+]-protocol ...</u>	all
Configure usable SSL protocol flavors for proxy usage	
<u>SSLProxyVerify level</u>	none
Type of remote server Certificate verification	
<u>SSLProxyVerifyDepth number</u>	1

SSLRandomSeed <i>context_source</i> [bytes]	
Pseudo Random Number Generator (PRNG) seeding source	
SSLRequire <i>expression</i>	
Allow access only when an arbitrarily complex boolean expression is	
SSLRequireSSL	
Deny access when SSL is not used for the HTTP request	
SSLSessionCache <i>type</i>	none
Type of the global/inter-process SSL Session Cache	
SSLSessionCacheTimeout <i>seconds</i>	300
Number of seconds before an SSL session expires in the Session C	
SSLUserName <i>varname</i>	
Variable name to determine user name	
SSLVerifyClient <i>level</i>	none
Type of Client Certificate verification	
SSLVerifyDepth <i>number</i>	1
Maximum depth of CA Certificates in Client Certificate verification	
StartServers <i>number</i>	
StartThreads <i>number</i>	
SuexecUserGroup <i>User Group</i>	
CGI	
ThreadLimit <i>number</i>	
ThreadsPerChild <i>number</i>	
ThreadStackSize <i>size</i>	
()	
TimeOut <i>seconds</i>	300

TraceEnable <i>[on off extended]</i> TRACE	on
TransferLog <i>file pipe</i>	
TypesConfig <i>file-path</i> mime.types	conf/mime.types
UnsetEnv <i>env-variable [env-variable] ...</i>	
UseCanonicalName On Off DNS	Off
UseCanonicalPhysicalPort On Off	Off
User <i>unix-userid</i>	#-1
UserDir <i>directory-filename</i>	
VirtualDocumentRoot <i>interpolated-directory none</i>	none
VirtualDocumentRootIP <i>interpolated-directory none</i> IP	none
<VirtualHost <i>addr[:port] [addr[:port]] ...> ...</i> </VirtualHost> IP	
VirtualScriptAlias <i>interpolated-directory none</i> CGI	none
VirtualScriptAliasIP <i>interpolated-directory none</i> IPCGI	none
Win32DisableAcceptEx	

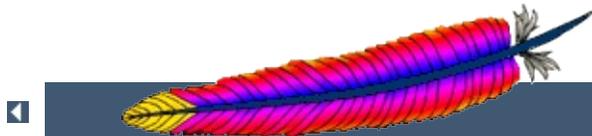
accept()AcceptEx()

[XBitHack on|off|full](#)

off

Parse SSI directives in files with the execute bit set

| | | |



Apache HTTP Server 2.2

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



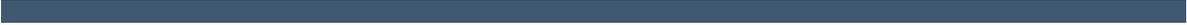
“Apache 2.0” Apache 2.2 []kajaa
biAji fei suncjs Daniel flytosea forehead

[LinuxFans.Org](#) sejishikong []

[LinuxSir.Org](#) bingzhou []

chm pdf





-
- [GPL](#)
-
-
- “” ///



-
- [Apache 2.2](#)
<http://lamp.linux.gov.cn/Apache/ApacheMenu/index.html>
 - [Apache 2.2](#)
<http://www.dogdoghome.com/lamp/Apache/ApacheMenu/index.html>

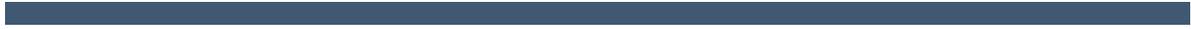
- [rarbz2zippdfchm](#)
- [rarbz2zippdfchm](#)



Apache“Apache2.0

QQ70171448 MSNcsfrank122@hotmail.com

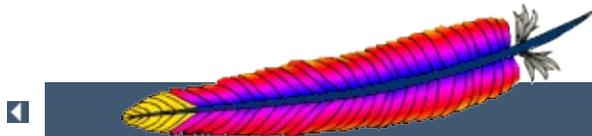




0

-
-





Apache HTTP Server 2.2

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

| | | 2006121 |

Apache



core

Apache HTTP

mpm_common

(MPM)

beos

BeOS(MPM)

event

workerMPM

mpm_network

Novell NetWare(MPM)

mpmt_os2

OS/2(MPM)

prefork

MPM

mpm_winnt

Windows NT/2000/XP/2003 MPM

worker

MPMMPM



A | C | D | E | F | H | I | L | M | N | P | R | S | U | V

[mod actions](#)

CGI

[mod alias](#)

URL

[mod asis](#)

HTTP

[mod auth basic](#)

[mod auth digest](#)

MD5()

[mod authn alias](#)

[mod authn anon](#)

[mod authn dbd](#)

SQL

[mod authn dbm](#)

DBM

[mod authn default](#)

[mod authn file](#)

[mod authnz ldap](#)

LDAP

[mod authz dbm](#)

DBM

[mod_authz_default](#)

[mod_authz_groupfile](#)

[mod_authz_host](#)

IP

[mod_authz_owner](#)

[mod_authz_user](#)

[mod_autoindex](#)

"ls""dir"

[mod_cache](#)

URI()

[mod_cern_meta](#)

ApacheCERN httpd

[mod_cgi](#)

MPM([prefork](#))CGI

[mod_cgid](#)

MPM([worker](#))CGICGI

[mod_charset_lite](#)

[mod_dav](#)

Apache[DAV](#)

[mod_dav_fs](#)

[mod_dav](#)

[mod_dav_lock](#)

[mod_dav](#)

[mod_dbd](#)

SQL

[mod_deflate](#)

[mod_dir](#)

....

[mod_disk_cache](#)

[mod_dumpio](#)

I/O

[mod_echo](#)

[mod_env](#)

ApacheCGISSI

[mod_example](#)

ApacheAPI

[mod_expires](#)

HTTP" Expires:"" Cache-Control:"

[mod_ext_filter](#)

[mod_file_cache](#)

Apache

[mod_filter](#)

[mod_headers](#)

HTTP

[mod_ident](#)

RFC1413ident

[mod_imagemap](#)

[mod_include](#)

(SSI)

[mod_info](#)

ApacheWeb

[mod_isapi](#)

WindowsISAPI

[mod_ldap](#)

LDAPLDAP

[mod_log_config](#)

[mod_log_forensic](#)

....

[mod_logio](#)

/HTTP

[mod_mem_cache](#)

[mod_mime](#)

(/)(MIME//)

[mod_mime_magic](#)

MIME

[mod_negotiation](#)

[mod_nw_ssl](#)

NetWareSSL

[mod_proxy](#)

HTTP/1.1/

[mod_proxy_ajp](#)

[mod_proxy](#)Apache JServ Protocol

[mod_proxy_balancer](#)

[mod_proxy](#)

[mod_proxy_connect](#)

[mod_proxy](#)HTTP CONNECT

[mod_proxy_ftp](#)

[mod_proxy](#)FTP

[mod_proxy_http](#)

[mod_proxy](#)HTTP

[mod_rewrite](#)

URL

[mod_setenvif](#)

[mod_so](#)

DSO

[mod_speling](#)

URL

[mod_ssl](#)

(SSL)(TLS)

[mod_status](#)

Web

[mod_suexec](#)

webCGISSI

[mod_unique_id](#)

[mod_userdir](#)

("/~username")

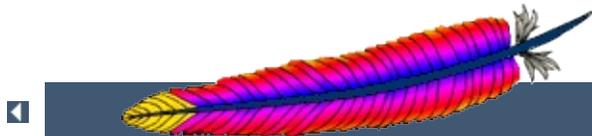
[mod_usertrack](#)

Session(Cookie)

[mod_version](#)

[mod vhost alias](#)





Apache HTTP Server 2.2

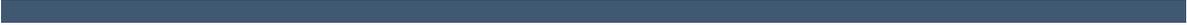
[Apache](#) > [HTTP Server](#) > > [2.2](#) >

| | | 200617 |

FAQApache< <http://httpd.apache.org/docs/2.2/faq/>>

[Apache 1.3 FAQ](#)





Apache HTTP Server



-
- [Apache](#)
 - [Apache HTTP Server](#)
 - [Apache](#)
 - [Apachelogo](#)

Apache

Apache(ASF)Apache

[Apache Software Foundation FAQ](#)

Apache HTTP Server(Apache httpd)ApacheHTTP(Web)
[About Apache](#)

Apache HTTP Server

- HTTP/1.1web
- HTTP/1.1(RFC2616)
-
- ApacheAPI
-
- Windows 2003/XP/2000/NT/9x Netware 5.x OS/2 Unix
-
- bug

Apache

ApacheApache HTTP Server 70%WWW24bug

Apachelogo

Apache

- Apacheweb'[Powered by Apache](#)'
- Apache ['Powered by Apache'](#) [Apachelogo](#) Apache



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

"....."

Apache

Apache()

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

[FAQ!](#)

ApacheApache

Apache bug

Apache[bug](#)bug () ""

Apache

[Freenode IRC#apache](#)

bug

httpd [bug](#)

dump [backtrace\(\)](#)

60Apache

Apache



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

Invalid argument: core_output_filter: writing data to the network

ApachesendfileApache sendfile

sendfile

[EnableSendfile](#)sendfile [EnableMMAP](#)

AcceptEx Failed

win32AcceptEx [Win32DisableAcceptEx](#)

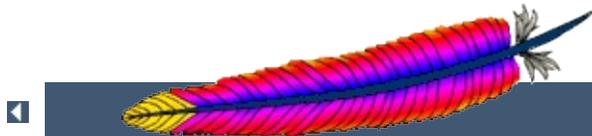
Premature end of script headers

CGI" Internal Server Error" [CGI](#)

Permission denied

error_log" Permission denied"" Forbidden"Apache
HTTP [UserGroup\(\)](#) chr

Fedora Core LinuxSELinux" Permission denied"
[Fedora SELinux FAQApache SELinux Policy Document](#)



Apache HTTP Server 2.2

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

| | | 200616 |



[Apache HTTP Server Version 2.2](#)



-
- - [1.3 2.0](#)
 - [2.0 2.2](#)
 - [Apache 2.1/2.2](#)
 - [Apache 2.0](#)
 - [Apache License](#)



- [Apache](#)

- [Apache](#)

-

-

- [DirectoryLocationFiles](#)

-

-

-

- [URL](#)

-

- [\(DSO\)](#)

-

-

- [Apache](#)

- [\(MPM\)](#)

- [Apache](#)

- [Apache](#)

-

- [suEXEC](#)

-

- [URL](#)



Apache

-
-
- [IP](#)
-
-
-
-
- [DNSApache](#)



Algebra

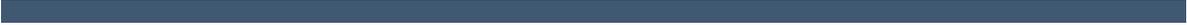
- (//)



SSL/TLS

-
- [SSL/TLS](#)
- [SSL/TLS](#)
- [SSL/TLS ...](#)
- [SSL/TLS](#)





-
-
- [CGI](#)
- [\(SSI\)](#)
- [.htaccess](#)
-





- [Microsoft WindowsApache](#)

- [Microsoft WindowsApache](#)

- [Novell NetWareApache](#)

- [HPUX](#)

- [ApacheEBCDIC](#)

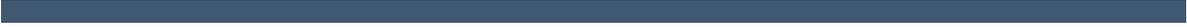


-

- [httpd](#)
- [ab](#)
- [apachectl](#)
- [apxs](#)
- [configure](#)
- [dbmmanage](#)
- [htcacheclean](#)
- [htdbm](#)
- [htdigest](#)
- [htpasswd](#)
- [logresolve](#)
- [rotatelogs](#)
- [suexec](#)

-





- [Apache](#)
- [Apache](#)

- [\(Core\)](#)
- [\(MPM\)](#)
- [beos\(MPM\)](#)
- [event\(MPM\)](#)
- [netware\(MPM\)](#)
- [os2\(MPM\)](#)
- [prefork\(MPM\)](#)
- [winnt\(MPM\)](#)
- [worker\(MPM\)](#)

- [mod_actions](#)
- [mod_alias](#)
- [mod_asis](#)
- [mod_auth_basic](#)
- [mod_auth_digest](#)
- [mod_authn_alias](#)
- [mod_authn_anon](#)
- [mod_authn_dbd](#)
- [mod_authn_dbm](#)
- [mod_authn_default](#)
- [mod_authn_file](#)
- [mod_authnz_ldap](#)
- [mod_authz_dbm](#)
- [mod_authz_default](#)
- [mod_authz_groupfile](#)
- [mod_authz_host](#)
- [mod_authz_owner](#)
- [mod_authz_user](#)
- [mod_autoindex](#)
- [mod_cache](#)

- [mod_cern_meta](#)
- [mod_cgi](#)
- [mod_cgid](#)
- [mod_charset_lite](#)
- [mod_dav](#)
- [mod_dav_fs](#)
- [mod_dav_lock](#)
- [mod_dbd](#)
- [mod_deflate](#)
- [mod_dir](#)
- [mod_disk_cache](#)
- [mod_dumpio](#)
- [mod_echo](#)
- [mod_env](#)
- [mod_example](#)
- [mod_expires](#)
- [mod_ext_filter](#)
- [mod_file_cache](#)
- [mod_filter](#)
- [mod_headers](#)
- [mod_ident](#)
- [mod_imagemap](#)
- [mod_include](#)
- [mod_info](#)
- [mod_isapi](#)
- [mod_ldap](#)
- [mod_log_config](#)
- [mod_log_forensic](#)
- [mod_logio](#)
- [mod_mem_cache](#)
- [mod_mime](#)
- [mod_mime_magic](#)
- [mod_negotiation](#)
- [mod_nw_ssl](#)

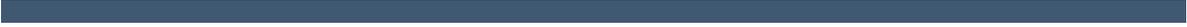
- [mod_proxy](#)
- [mod_proxy_ajp](#)
- [mod_proxy_balancer](#)
- [mod_proxy_connect](#)
- [mod_proxy_ftp](#)
- [mod_proxy_http](#)
- [mod_rewrite](#)
- [mod_setenvif](#)
- [mod_so](#)
- [mod_speling](#)
- [mod_ssl](#)
- [mod_status](#)
- [mod_suexec](#)
- [mod_unique_id](#)
- [mod_userdir](#)
- [mod_usertrack](#)
- [mod_version](#)
- [mod_vhost_alias](#)



-

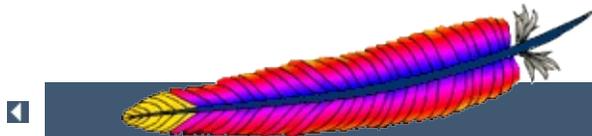
- [Apache API](#)
- [APR](#)
- [Apache2.0](#)
- [Apache2.0Hook](#)
- [Apache1.3Apache2.0](#)
- [Apache2.0](#)
- [Apache2.0](#)





-
-
-
-





◀ **Apache HTTP Server 2.2**

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

| | | 2006114 |

Apache HTTP



[httpd](#)

Apache

[apachectl](#)

Apache HTTP

[ab](#)

Apache HTTP

[apxs](#)

APache

[configure](#)

[dbmmanage](#)

DBM

[htcacheclean](#)

[htdigest](#)

[htdbm](#)

DBM

[htpasswd](#)

[htt2dbm](#)

RewriteMapdbm

[logresolve](#)

ApacheIP

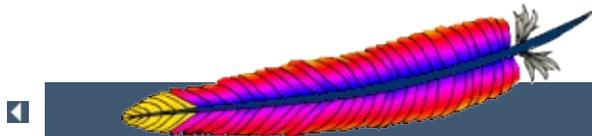
[rotatelogs](#)

Apache

[suexec](#)

Exec





Apache HTTP Server 2.2

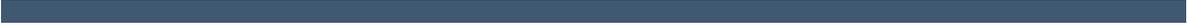
[Apache](#) > [HTTP Server](#) > > [2.2](#) > [SSL/TLS](#)

| | | 2006116 |

ApacheSSL/TLS

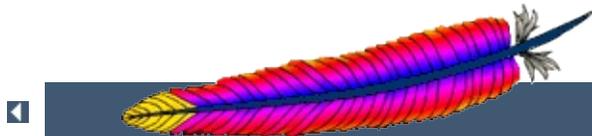
Apache HTTP [mod_ssl](#) (Secure Sockets Layer) (Transport Layer Security) [OpenSSL](#) Ralf S. Engelschall [mod_ssl](#)





[mod_ssl](#)





Apache HTTP Server 2.2

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

| | | 2006118 |

Apache

" "(www.company1.comwww.company2.com)IP" 
IP"IP" "

ApacheIP1.1IP" "" IP"

Apache1.3



-
- (IP)
 - IP(IP)
 -
 - ()
 -
 -



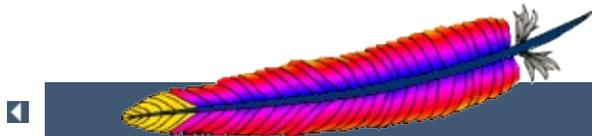
-
- <VirtualHost>
 - NameVirtualHost
 - ServerName
 - ServerAlias
 - ServerPath

Apache -S

```
/usr/local/apache2/bin/httpd -S
```

ApacheIP([httpd](#))

||||



◀ Apache HTTP Server 2.2

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

| | < > | ??? |

Developer Documentation for Apache 2.0

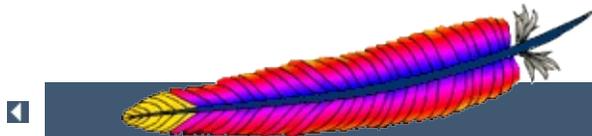
Many of the documents on these Developer pages are lifted from Apache 1.3's documentation. While they are all being updated to Apache 2.0, they are in different stages of progress. Please be patient, and point out any discrepancies or errors on the developer/ pages directly to the dev@httpd.apache.org mailing list.



-
- [Apache 1.3 API Notes](#)
 - [Apache 2.0 Hook Functions](#)
 - [Request Processing in Apache 2.0](#)
 - [How filters work in Apache 2.0](#)
 - [Converting Modules from Apache 1.3 to Apache 2.0](#)
 - [Debugging Memory Allocation in APR](#)
 - [Documenting Apache 2.0](#)
 - [Apache 2.0 Thread Safety Issues](#)



-
- Tools provided by Ian Holsman:
 - [Apache 2 cross reference](#)
 - [Autogenerated Apache 2 code documentation](#)
 - Module Development Tutorials by Kevin O'Donnell
 - [Integrating a module into the Apache build system](#)
 - [Handling configuration directives](#)
 - [Some notes on Apache module development by Ryan Bloom](#)
 - Developer articles at [apachetutor](#) include:
 - [Request Processing in Apache](#)
 - [Configuration for Modules](#)
 - [Resource Management in Apache](#)
 - [Connection Pooling in Apache](#)
 - [Introduction to Buckets and Brigades](#)



Apache HTTP Server 2.2

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

| | | 200619 |



Apache web

Apache HTTP2.2

Apache

Apache

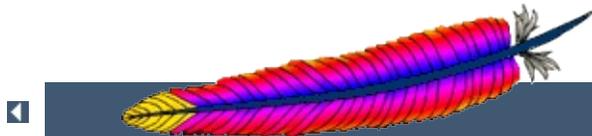
Apache web

URL

mod_rewrite mod_rewriteURL

Apache





Apache HTTP Server 2.2

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

| | | 2006114 |

httxt2dbm - RewriteMapdbm

httxt2dbmRewriteMapdbm(dbm)



```
httxt2dbm [ -v ] [ -f DBM_TYPE ] -i SOURCE_TXT -o  
OUTPUT_DBM
```



-v

-f

DBM APR
GDBM GDBM
SDBM SDBM
DB berkeley DB
NDBM NDBM
default

-i

dbm
key value
RewriteMap

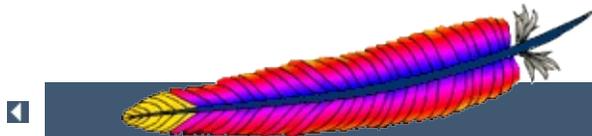
-o

dbm



```
httxt2dbm -i rewritesmap.txt -o rewritesmap.dbm  
httxt2dbm -f SDBM -i rewritesmap.txt -o  
rewritesmap.dbm
```

| | | |



Apache HTTP Server 2.2

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

| | | 2006112 |



Apache

WindowsApache2.0

[Microsoft WindowsApache](#)

Apache

WindowsApache

[Microsoft WindowsApache](#)



Novell NetWare

Novell NetWare 5.1Apache2.0

[Novell NetWareApache](#)

HP-UX

HP-UXApache

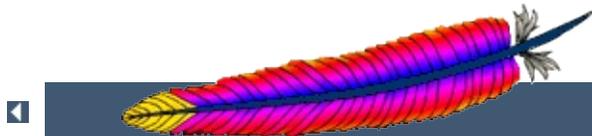
[HP-UXApache](#)

EBCDIC

Apache HTTP1.3EBCDICASCII

Apache HTTP2.0

[The Apache EBCDIC Port](#)



◀ **Apache HTTP Server 2.2**

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

| | | 2006114 |

suexec -

suexecApache HTTPCGI rootApache root

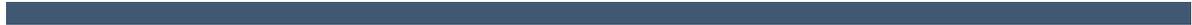
suexecroot setuid root

suexec [suexec](#))



suexec -V

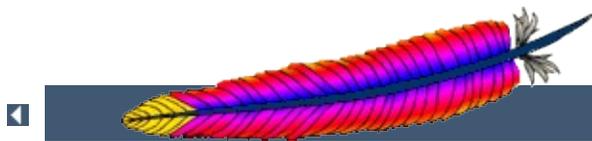




-v

root suexec

||||



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [.../](#)

| | | 200619 |

...



(Authentication)(Authorization)

CGI

CGI()webCGICGIApache webCGICGI

CGI

.htaccess

.htaccess("")

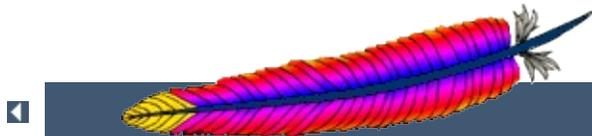
See: .htaccess

SSIHTMLHTMLCGI

See: (SSI)

UserDirURL http://example.com/~username/
" username" UserDir

See: (public_html)



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [URL](#)

| | < > | ??? |

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.

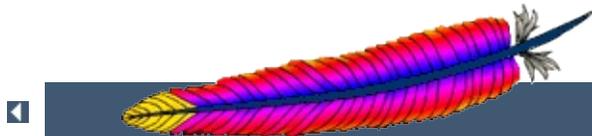
This module operates on the full URLs (including the path-info part) both in per-server context (`httpd.conf`) and per-directory context (`.htaccess`) and can even generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to 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.



-
- [mod_rewrite reference documentation](#)
 -
 - [Technical details](#)
 - [Practical solutions to common problems](#)
 - [Practical solutions to advanced problems](#)
 - [Glossary](#)



||||



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [URL](#)

||< >|???

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](#) [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.



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

And for a site running on port 80

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



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 an 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 an 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_F
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_F
RewriteRule  ^(.+)  /your/docroot/dir2/$1  [L]

# else go on for other Alias or ScriptAlias dire
# 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=([^/]+)/(.*) $1/$3 [E=STA
```



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\.[
RewriteRule ^(.+) %{HTTP_
RewriteRule ^www\.([\^.]+)\.host\.com(.*) /home/$
```



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`

Description:

By default, redirecting to an HTML anchor doesn't work, because `mod_rewrite` escapes the `#` character, turning it into `%23`. This, in turn, breaks the redirection.

Solution:

Use the `[NE]` flag on the `RewriteRule`. `NE` stands for No Escape.



Description:

When tricks like time-dependent content should happen a lot of webmasters still use CGI scripts which do for instance redirects to specialized pages. How can it be done via [mod_rewrite](#)?

Solution:

There are a lot of variables named TIME_XXX for rewrite conditions. In conjunction with the special lexicographic comparison patterns <STRING, >STRING=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.ht
```

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 HTML to XHTML 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$ $1 [C]
# rewrite to document.phtml if exists
RewriteCond %{REQUEST_FILENAME}.phtml -f
RewriteRule ^(.*)$ $1.phtml [S]
# else reverse the previous basename cutout
RewriteCond %{ENV:wasHTML} ^yes$
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=applicatio
```



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-
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  
RewriteRule .*\.gif$ -
```

```
RewriteCond %{HTTP_REFERER} !^$  
RewriteCond %{HTTP_REFERER} !.*\/foo-with-g  
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} ^badgu  
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      prg:/path/to/map.quux
RewriteRule   ^/~quux/(.*)$ /~quux/${quux-map:$1}
```

```
#!/path/to/perl

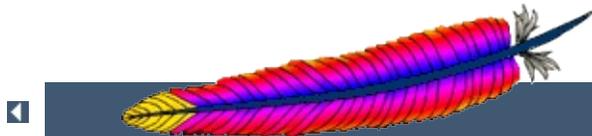
#  disable buffered I/O which would lead
#  to deadlocks for the Apache server
$| = 1;

#  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.





||||



Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [URL](#)

||< >|???

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 slightly change the examples for your situation, e.g. adding the [PT] flag when additionally using [mod_alias](#) [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.



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.us
```

```
RewriteMap      group-to-host     txt:/path/to/map.gr
```

```
RewriteMap      entity-to-host    txt:/path/to/map.en
```

```
RewriteRule     ^/u/([^/]+)/?(.*)  http://${user-to
```

```
RewriteRule     ^/g/([^/]+)/?(.*)  http://${group-to
```

```
RewriteRule     ^/e/([^/]+)/?(.*)  http://${entity-to
```

```
RewriteRule     ^/([uge])/([^/]+)/?$  /$1/$2
```

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



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/
```



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 Au
drwxrwxr-x  2 netsw  users    512 Jul  9 14:37 Be
drwxrwxr-x 12 netsw  users    512 Jul  9 00:34 Cr
drwxrwxr-x  5 netsw  users    512 Jul  9 00:41 Da
drwxrwxr-x  4 netsw  users    512 Jul 30 19:25 Di
drwxrwxr-x 10 netsw  users    512 Jul  9 01:54 Gr
drwxrwxr-x  5 netsw  users    512 Jul  9 01:58 Ha
drwxrwxr-x  8 netsw  users    512 Jul  9 03:19 In
drwxrwxr-x  3 netsw  users    512 Jul  9 03:21 Ma
drwxrwxr-x  3 netsw  users    512 Jul  9 03:24 Mi
drwxrwxr-x  9 netsw  users    512 Aug  1 16:33 Ne
drwxrwxr-x  2 netsw  users    512 Jul  9 05:53 Of
drwxrwxr-x  7 netsw  users    512 Jul  9 09:24 So
drwxrwxr-x  7 netsw  users    512 Jul  9 12:17 Sy
drwxrwxr-x 12 netsw  users    512 Aug  3 20:15 Ty
drwxrwxr-x 10 netsw  users    512 Jul  9 14:08 X1
```

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 .
drwxr-xr-x   18 netsw  users        512 Aug  5 15:51 D
-rw-rw-rw-    1 netsw  users    372982 Aug  5 16:35 L
-rw-r--r--    1 netsw  users        659 Aug  4 09:27 T
-rw-r--r--    1 netsw  users      5697 Aug  1 18:01 n
-rwxr-xr-x    1 netsw  users        579 Aug  2 10:33 n
-rwxr-xr-x    1 netsw  users      1532 Aug  1 17:35 n
-rwxr-xr-x    1 netsw  users      2866 Aug  5 14:49 n
drwxr-xr-x    2 netsw  users        512 Jul  8 23:47 n
-rwxr-xr-x    1 netsw  users     24050 Aug  5 15:49 n
-rwxr-xr-x    1 netsw  users      1589 Aug  3 18:43 n
-rwxr-xr-x    1 netsw  users      1885 Aug  1 17:41 n
-rw-r--r--    1 netsw  users        234 Jul 30 16:35 n
```

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 Mult
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.
RewriteRule     ^index\.html$                   netsw-home.

# strip out the subdirs when
# the browser requests us from perdir pages
RewriteRule     ^.+/(netsw-[^/]+/\.+)$          $1

# and now break the rewriting for local files
RewriteRule     ^netsw-home\.cgi.*              -
RewriteRule     ^netsw-changes\.cgi.*          -
RewriteRule     ^netsw-search\.cgi.*          -
RewriteRule     ^netsw-tree\.cgi$              -
RewriteRule     ^netsw-about\.html$            -
RewriteRule     ^netsw-img/\.*$                -

# anything else is a subdir which gets handled
# by another cgi script
RewriteRule     !^netsw-lsdir\.cgi.*          -
RewriteRule     (.*)                            netsw-lsdir
```

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



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} !-
RewriteRule  ^(.+) ht
```

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
```

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.



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/t
RewriteRule   ^/CxAN/(.*)       %{REMOTE_HO
RewriteRule   ^.+\.([a-zA-Z]+)::(.*)$  ${multiplex
```

```
##
##  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 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

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

RewriteRule ^foo\.html$ foo.32.html
```

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.tstimpr
```

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^usa-news\.html$ http://www.quux-
```

Reverse Dynamic Mirror

Description:

...

Solution:

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

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 w
DENY   Host *                Port *      --> Host w
```

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/.ww
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^/home/([^/]+)/.www/?(.*) http://www2.
```

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-www6` - 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 webservers.

2. DNS Load-Balancing

A sophisticated DNS-based method for load-balancing is to use the program `lbname`d which can be found at <http://www.stanford.edu/~schemers/docs/lbname/d/lbname/d.html>. 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
$last     = 5;          # the last server in the r
$domain   = "foo.dom"; # the domainname

$cnt = 0;
while (<STDIN>) {
    $cnt = (($cnt+1) % ($last+1-$first));
    $server = sprintf("%s%d.%s", $name, $cnt+$f
    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_INFOQUERY_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 `cgwrap` 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 `cgwrap` needs the URL in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uqe]/([^/]+)/\.www/(.+)\.scgi(.*).  
... /internal/cgi/user/cgwrap/~$1/$2.scgi$3 [NS,
```

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 `swidx` program from within `/u/user/foo/` via hyperlink to

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

which is ugly. Because we have to hard-code **both** the location of the area 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 ^/([uqe])/([^/]+)(/?.*)/\*$ /interna
RewriteRule ^/([uqe])/([^/]+)(/?.*):log /interna
```

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 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=ap
```

Here a request to `page.html` leads to an internal run of a corresponding `page.cgi` if `page.html` is still missing or has a file size of 0. 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 of 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 web browser 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 ^(/[uqe]/[^/]+/?.*):refresh /intern
```

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
```

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 refresh
##  Copyright (c) 1997 Ralf S. Engelschall, All Ri
##
$| = 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;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C"
eval "\$$name = \"$value\"";
}
$QS_s = 1 if ($QS_s eq "");
$QS_n = 3600 if ($QS_n eq "");
if ($QS_f eq "") {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "<b>ERROR</b>: No file given
    exit(0);
}
if (! -f $QS_f) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "<b>ERROR</b>: File $QS_f no
    exit(0);
}

sub print_http_headers_multipart_begin {
```

```

print "HTTP/1.0 200 OK\n";
$bound = "ThisRandomString12345";
print "Content-type: multipart/x-mixed-replace
&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, $size) = stat($fi
    $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) =
    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:

[<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-forma
CustomLog /path/to/access_log "%{VHOST}e %h %l %
:

#  enable the rewriting engine in the main server
RewriteEngine on

#  define two maps: one for fixing the URL and on
#  the available virtual hosts with their corresp
#  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 1
#       and remember the virtual host for logging p
RewriteRule     ^(/. *)$        %1/$1 [E=VHOST:${lowercas
:

```



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-FOUN
RewriteCond   ${hosts-deny:%{REMOTE_ADDR}}|NOT-FOUN
RewriteRule   ^/.*      -    [F]
```

For Apache <= 1.3b6:

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

```
##
##  hosts.deny
##
##  ATTENTION! This is a map, not a list, even whe
##              mod_rewrite parses it for key/value
##              dummy value "-" must be present for
##
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} ^badgu  
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_basic`).

Solution:

We use a list of rewrite conditions to exclude all except our friends:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^frien
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^frien
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^frien
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]

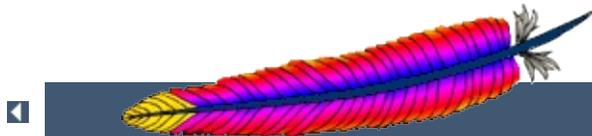
RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}|NOT-FOUND}
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

... 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://so
```

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).



||||



Apache HTTP Server 2.2

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

||< >|???

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 conditions that may only show up in certain conditions under heavy load.



Global and Static Variables

When writing your module or when trying to determine if a module or 3rd party library is thread safe there are some common things to keep in mind.

First, you need to recognize that in a threaded model each individual thread has its own program counter, stack and registers. Local variables live on the stack, so those are fine. You need to watch out for any static or global variables. This doesn't mean that you are absolutely not allowed to use static or global variables. There are times when you actually want something to affect all threads, but generally you need to avoid using them if you want your code to be thread safe.

In the case where you have a global variable that needs to be global and accessed by all threads, be very careful when you update it. If, for example, it is an incrementing counter, you need to atomically increment it to avoid race conditions with other threads. You do this using a mutex (mutual exclusion). Lock the mutex, read the current value, increment it and write it back and then unlock the mutex. Any other thread that wants to modify the value has to first check the mutex and block until it is cleared.

If you are using [APR](#), have a look at the `apr_atomic_*` functions and the `apr_thread_mutex_*` functions.



This is a common global variable that holds the error number of the last error that occurred. If one thread calls a low-level function that sets `errno` and then another thread checks it, we are bleeding error numbers from one thread into another. To solve this, make sure your module or library defines `_REENTRANT` or is compiled with `-D_REENTRANT`. This will make `errno` a per-thread variable and should hopefully be transparent to the code. It does this by doing something like this:

```
#define errno (*(__errno_location()))
```

which means that accessing `errno` will call `__errno_location()` which is provided by the `libc`. Setting `_REENTRANT` also forces redefinition of some other functions to their `*_r` equivalents and sometimes changes the common `getc/putc` macros into safer function calls. Check your `libc` documentation for specifics. Instead of, or in addition to `_REENTRANT` the symbols that may affect this are `_POSIX_C_SOURCE`, `_THREAD_SAFE`, `_SVID_SOURCE`, and `_BSD_SOURCE`.



Not only do things have to be thread safe, but they also have to be reentrant. `strtok()` is an obvious one. You call it the first time with your delimiter which it then remembers and on each subsequent call it returns the next token. Obviously if multiple threads are calling it you will have a problem. Most systems have a reentrant version of the function called `strtok_r()` where you pass in an extra argument which contains an allocated `char *` which the function will use instead of its own static storage for maintaining the tokenizing state. If you are using [APR](#) you can use `apr_strtok()`.

`crypt()` is another function that tends to not be reentrant, so if you run across calls to that function in a library, watch out. On some systems it is reentrant though, so it is not always a problem. If your system has `crypt_r()` chances are you should be using that, or if possible simply avoid the whole mess by using md5 instead.



The following is a list of common libraries that are used by 3rd party Apache modules. You can check to see if your module is using a potentially unsafe library by using tools such as `ldd(1)` `nm(1)`. For [PHP](#), for example, try this:

```
% ldd libphp4.so
libsablot.so.0 => /usr/local/lib/libsablot.so.0
(0x401f6000)
libexpat.so.0 => /usr/lib/libexpat.so.0
(0x402da000)
libsnmp.so.0 => /usr/lib/libsnmp.so.0 (0x402f9000)
libpdf.so.1 => /usr/local/lib/libpdf.so.1
(0x40353000)
libz.so.1 => /usr/lib/libz.so.1 (0x403e2000)
libpng.so.2 => /usr/lib/libpng.so.2 (0x403f0000)
libmysqlclient.so.11 =>
/usr/lib/libmysqlclient.so.11 (0x40411000)
libming.so => /usr/lib/libming.so (0x40449000)
libm.so.6 => /lib/libm.so.6 (0x40487000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6
(0x404a8000)
libjpeg.so.62 => /usr/lib/libjpeg.so.62
(0x404e7000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40505000)
libssl.so.2 => /lib/libssl.so.2 (0x40532000)
libcrypto.so.2 => /lib/libcrypto.so.2 (0x40560000)
libresolv.so.2 => /lib/libresolv.so.2 (0x40624000)
libdl.so.2 => /lib/libdl.so.2 (0x40634000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40637000)
libc.so.6 => /lib/libc.so.6 (0x4064b000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2
(0x80000000)
```

In addition to these libraries you will need to have a look at any libraries linked statically into the module. You can use `nm(1)` to look

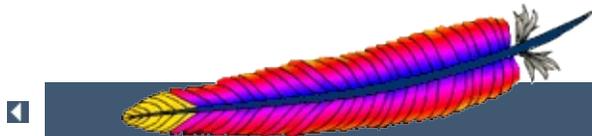
for individual symbols in the module.



Please drop a note to dev@httpd.apache.org if you have additions or corrections to this list.

Library	Version	Thread Safe?	Notes
ASpell/PSpell		?	
Berkeley DB	3.x, 4.x	Yes	Be careful about sharing a connection across threads.
bzip2		Yes	Both low-level and high-level APIs are thread-safe. However, high-level API requires thread-local errno.
cdb		?	
C-Client		Perhaps	c-client uses <code>strtok()</code> <code>gethostbyname()</code> which are not thread-safe on most C library implementations. c-client's static data is shared across threads. If <code>strtok()</code> <code>gethostbyname()</code> are thread-safe on your platform, c-client <i>may</i> be thread-safe.
cpdflib		?	
libcrypt		?	
Expat		Yes	Need a separate parser instance per thread.
FreeTDS		?	
FreeType		?	
GD 1.8.x		?	
GD 2.0.x		?	
gdbm		No	Errors returned via a static <code>gdbm_error</code> .
ImageMagick	5.2.2	Yes	ImageMagick docs claim it is thread safe in version 5.2.2 (see Change log).
Imlib2		?	
libjpeg	v6b	?	

libmysqlclient		Yes	Use mysqlclient_r library variant to ensure safety. For more information, please read http://www.mysql.com/doc/en/Threaded
Ming	0.2a	?	
Net-SNMP	5.0.x	?	
OpenLDAP	2.1.x	Yes	Use ldap_r library variant to ensure th
OpenSSL	0.9.6g	Yes	Requires proper usage of CRYPTO_num CRYPTO_set_locking_callback, CRYPTO_set_id_callback
liboci8 (Oracle 8+)	8.x,9.x	?	
pdflib	5.0.x	Yes	PDFLib docs claim it is thread safe; cha indicates it has been partially thread-sa V1.91: http://www.pdflib.com/products/pdflib/in
libpng	1.0.x	?	
libpng	1.2.x	?	
libpq (PostgreSQL)	7.x	Yes	Don't share connections across threads out for crypt () calls
Sablotron	0.95	?	
zlib	1.1.4	Yes	Relies upon thread-safe zalloc and zfre Default is to use libc's calloc/free which safe.



||||

◀ Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [URL](#)

|| < > | ??? |

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.



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.



`mod_rewrite` uses the [Perl Compatible Regular Expression](#) vocabulary. In this document, we do not attempt to provide a detailed reference to regular expressions. For that, we recommend the [PCRE man pages](#), the [Perl regular expression man page](#), and [Mastering Regular Expressions, by Jeffrey Friedl](#).

In this document, we attempt to provide enough of a regex vocabulary to get you started, without being overwhelming, in the hope that [RewriteRules](#) will be scientific formulae, rather than magical incantations.

Regex vocabulary

The following are the minimal building blocks you will need, in order to write regular expressions and [RewriteRules](#).

Character	Meaning
.	Matches any character

Regex Back-Reference Availability

One important thing here has to be remembered: Whenever you use parentheses in *Pattern* or in one of the *CondPattern*, back-references are internally created which can be used with the strings `$N%N` (see below). These are available for creating the strings *Substitution* *TestString*. Figure 2 shows to which locations the back-references are transferred for expansion.

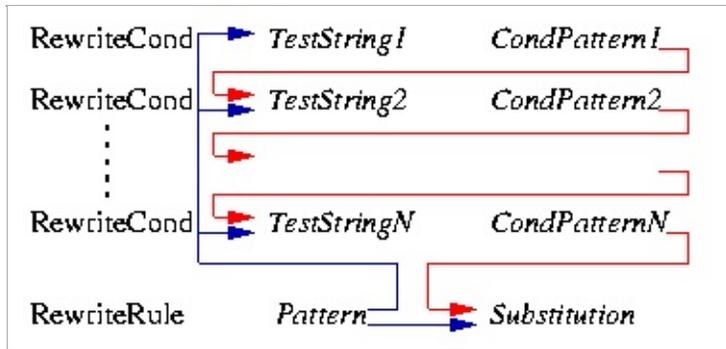


Figure 2: The back-

reference flow through a rule.



REWRITERULE BASICS

Basic anatomy of a RewriteRule, with exhaustively annotated simple examples.



RewriteRule

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.

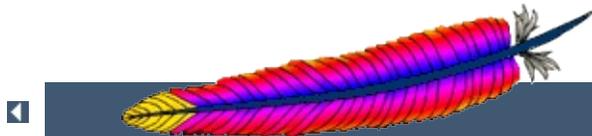


This module keeps track of two additional (non-standard) CGI/SSI environment variables named SCRIPT_URLSCRIPT_URI. These contain the *logical* Web-view to the current resource, while the standard CGI/SSI variables SCRIPT_NAMESCRIPT_FILENAME contain the *physical* System-view.

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.l  
SCRIPT_FILENAME=/u/rse/.www/index.html  
SCRIPT_URL=/u/rse/  
SCRIPT_URI=http://en1.engelschall.com/u/rse/
```



||||

◀ Apache HTTP Server 2.2

[Apache](#) > [HTTP Server](#) > > [2.2](#) > [URL](#)

||< >|???

Apache mod_rewrite Technical Details

This document discusses some of the technical details of mod_rewrite and URL matching.



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.



API PHASES

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!



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.

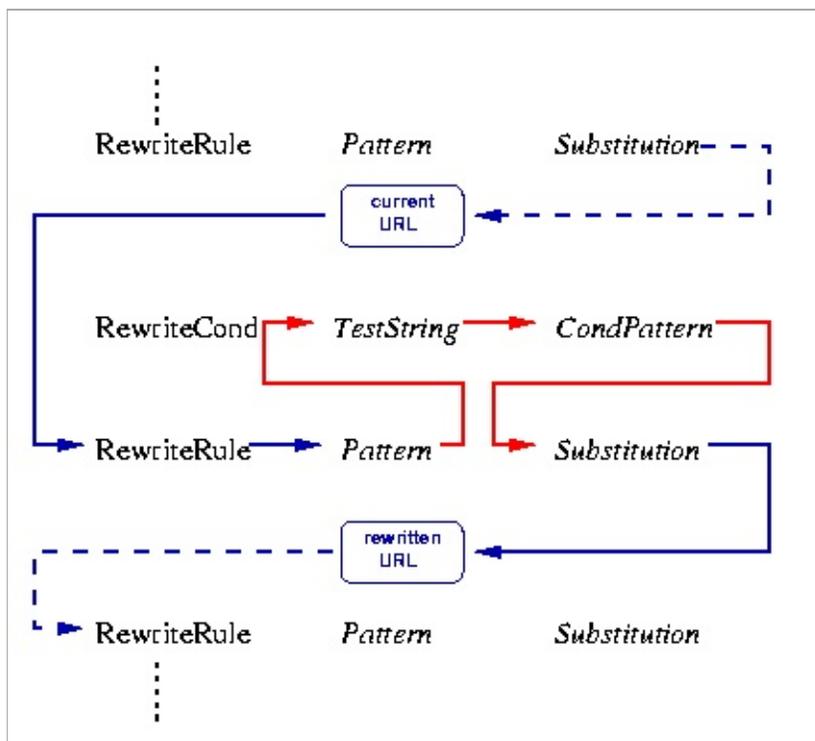


Figure 1:The

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*.