Apache HTTP 2.2

Google
(DSO)

URL

SSL/TLS

CGI Suexec

URL Rewriting
How-To /

CGI:
.htaccess
Server Side Includes (SSI)
____________________(public_html)
Microsoft Windows
Novell NetWare
EBCDIC
Apache

2.0 2.2

2.0 2.2 1.3

Apache 2.2
2.0

- mod_imap  mod_imagemap
- mod_auth  mod_auth_basic  mod_authn_file,  mod_authz_user  mod_authz_groupfile
- mod_access  mod_authz_host
- mod_authz_ldap  mod_authnz_ldap
- APR 1.0 API
- PCRE 5.0
2.0

2.2

conf/original

:

- `apachectl startssl SSL`
- `mod_ssl apachectl startssl SSL`
- `conf/extra/httpd-ssl.conf`
- `UseCanonicalName Off`
- `UseCanonicalName On`
- `mod_userdir UserDir public_html`
- `mod_auth_digest AuthDigestFile AuthUserFile mod_authn_file`
- Apache 2.0  mod_cache
- Apache 2.0  mod_disk_cache
- Apache 2.0  mod_mem_cache
- Apache 2.0  mod_charset_lite
- Apache 2.0  mod_dumpio
Apache 2.2

Apache HTTP 2.0 2.2

1.3
Authn/Authz

mod_authn_alias

mod_cache  mod_disk_cache  mod_mem_cache

htcacheclean

Apache

prefork  worker  event  MPM

graceful-stop

httpd

mod_proxy_balancer  mod_proxy

mod_proxy_ajp  Apache Tomcat

Apache JS

Protocol version 1.3

Perl Compatible Regular Expression Library (PCRE: Perl )

5.0  httpd configure  --with-pcre

PCRE

mod_filter

2.0

httpd  32-bit Unix  2GB

Event MPM

event  MPM  Keep Alive

Keep Alive

SQL

mod_dbd  apr_dbd  SQL
Authn/Authz

aaa

mod_authn_file mod_auth_dbm
mod_authn_dbm mod_access mod_authz
mod_authn_alias

mod_authnz_ldap

2.0 mod_auth_ldap 2.2 Authn/Authz

Require

mod_authz_owner

mod_version

mod_info

?config Apache

mod_ssl

RFC 2817 TLS

mod_imagemap

mod_imap mod_imagemap
httpd
  -M

httxt2dbm
dbm
**APR 1.0 API**

Apache 2.2 APR 1.0 API

**APR Web**

**Authn/Authz**

- `mod_auth_*` -> HTTP
- `mod_authn_*` ->
- `mod_authz_*` ->
- `mod_authnz_*` ->

`ap_log_cerror`

`test_config` -t `httpd`

**MPM's**

`ThreadStackSize` MPM

`ap_register_output_filter_protocol`

`ap_filter_protocol` `mod_filter`

(root)

**API**

`pcreposix.h; ap_regex.h`

`regex.h ap_regex.h ap_regexec` `ap_regcomp, ap_regexexec`

**DBD (SQL API)**

Apache 1.x 2.0 SQL
Apache 2.1       ap_dbd API
APR 1.2         apr_dbd API

SQL
Apache 2.0

Apache HTTP 1.3 2.0
Unix
  POSIX Unix
  autoconf  libtool

Apache
  configure

Unix
  Apache 2.0  BeOSOS/2Windows  Unix
  (MPM)  Apache Portable Runtime (APR)  API
  POSIX

Apache API
  2.0  API  1.3 /

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

Apache
  Server Side Include

SSI

vhost

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

Apache 2.0  Perl (PCRE)  Perl
mod_ssl
Apache 2.0 OpenSSL SSL/TLS

mod_dav
Apache 2.0 Versioning (DAV)

mod_deflate
Apache 2.0

mod_auth_ldap
Apache 2.0.41 HTTP L

mod_auth_digest

mod_charset_lite
Apache 2.0

mod_file_cache
Apache 2.0 Apache 1.3 m

mod_headers
Apache 2.0

mod_proxy
proxy HTTP/1.1 proxy ()
proxy_connect, proxy_ftp, proxy_http

mod_negotiation
NOT ACCEPTABLE MULTIPLE CHOICES ForceLanguagePriority

mod_autoindex
Autoindex HTML
<table>
<thead>
<tr>
<th><strong>mod_include</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSI</td>
</tr>
<tr>
<td><strong>mod_include</strong> $0 .. $9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>mod_auth_dbm</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AuthDBMType</strong> DBM</td>
</tr>
</tbody>
</table>
The Apache License, Version 2.0

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

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions

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

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

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

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

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

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not
limited to compiled object code, generated documentation, and conversions to other media types.

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

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

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

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by
Licensor and subsequently incorporated within the Work.

2. **Grant of Copyright License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. **Grant of Patent License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. **Redistribution.** You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   a. You must give any other recipients of the Work or Derivative Works a copy of this License; and

   b. You must cause any modified files to carry prominent notices stating that You changed the files; and
c. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

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

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

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

6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

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

END OF TERMS AND CONDITIONS

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

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

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

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

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

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

Copyright 2017 The Apache Software Foundation.
Licensed under the Apache License, Version 2.0.
Apache HTTP 2.2
<table>
<thead>
<tr>
<th>Unix</th>
<th>Unix</th>
<th>Apache HTTPd</th>
<th>Windows</th>
<th>Apache HTTPd</th>
<th>Open Source</th>
<th>li</th>
<th>(2.2.50 2.2.51)</th>
</tr>
</thead>
</table>
$ lynx http://httpd.apache.org/download.cgi
$ gzip -d httpd-NN.tar.gz
$ tar xvf httpd-NN.tar
$ cd httpd-NN
$ ./configure --prefix=PREFIX
$ make
$ make install
$ vi PREFIX/conf/httpd.conf
$ PREFIX/bin/apachectl -k start

NN          PREFIX
/usr/local/apache2

Apache HTTP
Apache HTTPd:

50 MB  Apache 10!

ANSI-C

ANSI-C  Free Software Foundation (FSF)  GNU C
compiler (GCC)  GCC  ANSI
make

HTTP
xntpd  NTP  NTP

Perl 5 []

  apxs  dbmmanage  Perl Perl
  ) Perl ( configure

Apache HTTPd

apr/apr-util >= 1.2

apr  apr-util  Apache HTTPd
util 1.0 1.1

apr/apr-util :

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

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

# Configure httpd
cd ../../*
./configure --with-apr=/usr/local/apr-httpd/ --with-apr-util=/usr/local/apr-util-httpd/
Apache HTTP
HTTPd
INSTALL.bindist

Apache HTTP
Apache HTTPd tarball

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

autoconf libtool buildconfig

./configure

Apache HTTPd

--prefix

Apache HTTPd

--enable-module

Apache HTTPd

--enable-module=shared

disable-module Base

configure
configure

configure

/configure

Makefile

/configure

/configure

/configure

/configure

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

/configure

/configure

/configure
Apache HTTPd

$ make
<table>
<thead>
<tr>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ make install</td>
</tr>
</tbody>
</table>

```bash
 PREFIX (  --prefix )
```

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ make install</td>
</tr>
</tbody>
</table>
$ vi PREFIX/conf/httpd.conf

docs/manual/ Apache HTTP
http://httpd.apache.org/docs/2.2/
Apache HTTP:

$ PREFIX/bin/apachectl -k start

URL http://localhost/

PREFIX/htdocs/:

$ PREFIX/bin/apachectl -k stop
API

(2.2.55 2.2.57)

configure

config.nice

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

Windows Apache
ME

Unix  httpd

httpd
apachectl
Listen  80 (1024)

```
httpd
apachectl
HTTPD
httpd
httpd.conf
```

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

... 

- root 
- Apache 

Apache FAQ
root

apachectl  SysV init
httpd  init
<table>
<thead>
<tr>
<th>Unix</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 9x, ME</td>
<td>Apache</td>
</tr>
</tbody>
</table>

- `httpd`
- `apachectl`
Apache

USR1:

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

httpd 2 -k graceful-stop

httpd:

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

ServerRoot  PidFile
TERM
    apachectl -k stop
TERM    stop kill
Graceful Restart

USR1
    apachectl -k graceful

USR1  graceful

MPM

mod_status

USR1

status

USR1

((httpd)

httpd)
: HUP
    apachectl -k restart

HUP restart TERM kill

mod_status HUP
Signal: WINCH
  apachectl -k graceful-stop

  WINCH  graceful-stop
Listen
  TERM

TERM "graceful"
httpd

graceful-stop ()
Lockfile  ScriptSock
CGI

PID  rotatelogs
Apache HTTP
mod_mime

<IfDefine>
 Include
  TypesConfig
</IfDefine>

Apache
Include

MIME
Apache 1 1

"\"

"#"

apachectl configtest
mod so  <IfModule>
        LoadModule
    
Apache
LoadModule
    -l
<Directory>
<DirectoryMatch>
<Files>
<FilesMatch>
<Location>
<LocationMatch>
<VirtualHost>

<FilesMatch>, <Location>, <LocationMatch>

Apache
Apache

.htaccess

.htaccess

.htaccess
URL .htaccess
<IfDefine> . <IfModule> .

<IfVersion>

<IfDefine> httpd httpd -DClosedForNow :

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

<IfModule>

LoadModule

mod_mime_magic MimeMagicFiles

<IfModule mod_mime_magic.c>
MimeMagicFile conf/magic
</IfModule>
<IfVersion>  <IfDefine>  <IfModule>
httpd

<IfVersion >= 2.1>
   # this happens only in versions greater or
   # equal 2.1.0.
</IfVersion>

<IfDefine>,  <IfModule>,  <IfVersion> "!"
Apache Unix
/usr/local/apache2

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

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

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

<Files> private.html

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

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

<Directory /var/web/dir1>
<Files private.html>
Order allow,deny
Deny from all
</Files>
</Directory>
<Location> /private URL
http://yoursite.example.com/private,
http://yoursite.example.com/private123,
http://yoursite.example.com/private/dir/file.html
/private

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

<Location> URL mod_status Apache
server-status

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

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

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

regex

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

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

<Directory>  <Files> ()

<Location>
  <Location> (URL)

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

http://yoursite.example.com/dir/ ?
http://yoursite.example.com/DIR/
  (Options)

  <Location /> URL
<VirtualHost>
<ProxyMatch>
  URL
cnn.com
</ProxyMatch>

<Proxy http://cnn.com/*>
Order allow,deny
Deny from all
</Proxy>
Context  <Directory>  <DirectoryMatch>,  
<Files>, <FilesMatch>, <Location>, <LocationMatch>,  
<Proxy>, <ProxyMatch>

- AllowOverride  <Directory>
- FollowSymLinks  SymLinksIfOwnerMatch  Options  
  <Directory>  .htaccess
- Options  <Files>  <FilesMatch>
1. <Directory> () .htaccess ( .htaccess <Directory> )
2. <DirectoryMatch> ( <Directory ->
3. <Files> <FilesMatch>
4. <Location> <LocationMatch>

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

<VirtualHost>
mod_proxy <Directory>

(URL Alias
<Location>/<LocationMatch>

A > B > C > D > E

<Location /> E </Location>

<Files f.html> D </Files>
<VirtualHost *>
  <Directory /a/b>
    B
  </Directory>
</VirtualHost>

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

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

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

  # Woops! This <Directory> section will have no effect
  <Directory />
  Order allow,deny
  Allow from all
  Deny from badguy.example.com
</Directory>
This document supplements the mod_cache, mod_disk_cache, mod_mem_cache, mod_file_cache and htcacheclean reference documentation. It describes how to use Apache's caching features to accelerate web and proxy serving, while avoiding common problems and misconfigurations.
As of Apache HTTP server version 2.2 mod_cache and mod_file_cache are no longer marked experimental and are considered suitable for production use. These caching architectures provide a powerful means to accelerate HTTP handling, both as an origin webserver and as a proxy.

mod_cache and its provider modules mod_mem_cache and mod_disk_cache provide intelligent, HTTP-aware caching. The content itself is stored in the cache, and mod_cache aims to honour all of the various HTTP headers and options that control the cachability of content. It can handle both local and proxied content. mod_cache is aimed at both simple and complex caching configurations, where you are dealing with proxied content, dynamic local content or have a need to speed up access to local files which change with time.

mod_file_cache on the other hand presents a more basic, but sometimes useful, form of caching. Rather than maintain the complexity of actively ensuring the cachability of URLs, mod_file_cache offers file-handle and memory-mapping tricks to keep a cache of files as they were when Apache was last started. As such, mod_file_cache is aimed at improving the access time to local static files which do not change very often.

As mod_file_cache presents a relatively simple caching implementation, apart from the specific sections on CacheFile and MMapFile, the explanations in this guide cover the mod_cache caching architecture.

To get the most from this document, you should be familiar with the basics of HTTP, and have read the Users’ Guides to Mapping URLs to the Filesystem and Content negotiation.
There are two main stages in `mod_cache` that can occur in the lifetime of a request. First, `mod_cache` is a URL mapping module, which means that if a URL has been cached, and the cached version of that URL has not expired, the request will be served directly by `mod_cache`.

This means that any other stages that might ordinarily happen in the process of serving a request -- for example being handled by `mod_proxy`, or `mod_rewrite` -- won't happen. But then this is the point of caching content in the first place.

If the URL is not found within the cache, `mod_cache` will add a filter to the request handling. After Apache has located the content by the usual means, the filter will be run as the content is served. If the content is determined to be cacheable, the content will be saved to the cache for future serving.

If the URL is found within the cache, but also found to have expired, the filter is added anyway, but `mod_cache` will create a conditional request to the backend, to determine if the cached version is still current. If the cached version is still current, its meta-information will be updated and the request will be served from the cache. If the cached version is no longer current, the cached version will be deleted and the filter will save the updated
content to the cache as it is served.

**Improving Cache Hits**

When caching locally generated content, ensuring that `UseCanonicalName` is set to On can dramatically improve the ratio of cache hits. This is because the hostname of the virtual-host serving the content forms a part of the cache key. With the setting set to On virtual-hosts with multiple server names or aliases will not produce differently cached entities, and instead content will be cached as per the canonical hostname.

Because caching is performed within the URL to filename translation phase, cached documents will only be served in response to URL requests. Ordinarily this is of little consequence, but there is one circumstance in which it matters: If you are using **Server Side Includes**:

```html
<!-- The following include can be cached -->
<!--#include virtual="/footer.html" -->

<!-- The following include can not be cached -->
<!--#include file="/path/to/footer.html" -->
```

If you are using Server Side Includes, and want the benefit of speedy serves from the cache, you should use virtual include types.

**Expiry Periods**

The default expiry period for cached entities is one hour, however this can be easily over-ridden by using the `CacheDefaultExpire` directive. This default is only used when the original source of the content does not specify an expire time or time of last modification.
If a response does not include an Expires header but does include a Last-Modified header, mod_cache can infer an expiry period based on the use of the CacheLastModifiedFactor directive.

For local content, mod_expires may be used to fine-tune the expiry period.

The maximum expiry period may also be controlled by using the CacheMaxExpire.

A Brief Guide to Conditional Requests

When content expires from the cache and is re-requested from the backend or content provider, rather than pass on the original request, Apache will use a conditional request instead.

HTTP offers a number of headers which allow a client, or cache to discern between different versions of the same content. For example if a resource was served with an "Etag:" header, it is possible to make a conditional request with an "If-None-Match:" header. If a resource was served with a "Last-Modified:" header it is possible to make a conditional request with an "If-Modified-Since:" header, and so on.

When such a conditional request is made, the response differs depending on whether the content matches the conditions. If a request is made with an "If-Modified-Since:" header, and the content has not been modified since the time indicated in the request then a terse "304 Not Modified" response is issued.

If the content has changed, then it is served as if the request were not conditional to begin with.

The benefits of conditional requests in relation to caching are twofold. Firstly, when making such a request to the backend, if the
content from the backend matches the content in the store, this can be determined easily and without the overhead of transferring the entire resource.

Secondly, conditional requests are usually less strenuous on the backend. For static files, typically all that is involved is a call to `stat()` or similar system call, to see if the file has changed in size or modification time. As such, even if Apache is caching local content, even expired content may still be served faster from the cache if it has not changed. As long as reading from the cache store is faster than reading from the backend (e.g. an in-memory cache compared to reading from disk).

**What Can be Cached?**

As mentioned already, the two styles of caching in Apache work differently, `mod_file_cache` caching maintains file contents as they were when Apache was started. When a request is made for a file that is cached by this module, it is intercepted and the cached file is served.

`mod_cache` caching on the other hand is more complex. When serving a request, if it has not been cached previously, the caching module will determine if the content is cacheable. The conditions for determining cachability of a response are;

1. Caching must be enabled for this URL. See the `CacheEnable` and `CacheDisable` directives.

2. The response must have a HTTP status code of 200, 203, 300, 301 or 410.

3. The request must be a HTTP GET request.

4. If the request contains an "Authorization:" header, the response will not be cached.
5. If the response contains an "Authorization:" header, it must also contain an "s-maxage", "must-revalidate" or "public" option in the "Cache-Control:" header.

6. If the URL included a query string (e.g. from a HTML form GET method) it will not be cached unless the response specifies an explicit expiration by including an "Expires:" header or the max-age or s-maxage directive of the "Cache-Control:" header, as per RFC2616 sections 13.9 and 13.2.1.

7. If the response has a status of 200 (OK), the response must also include at least one of the "Etag", "Last-Modified" or the "Expires" headers, or the max-age or s-maxage directive of the "Cache-Control:" header, unless the CacheIgnoreNoLastMod directive has been used to require otherwise.

8. If the response includes the "private" option in a "Cache-Control:" header, it will not be stored unless the CacheStorePrivate has been used to require otherwise.

9. Likewise, if the response includes the "no-store" option in a "Cache-Control:" header, it will not be stored unless the CacheStoreNoStore has been used.

10. A response will not be stored if it includes a "Vary:" header containing the match-all "*".

What Should Not be Cached?

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.

If you have dynamic content which changes depending on the IP address of the requester, or changes every 5 minutes, it should almost certainly not be cached.
If on the other hand, the content served differs depending on the values of various HTTP headers, it might be possible to cache it intelligently through the use of a "Vary" header.

**Variable/Negotiated Content**

If a response with a "Vary" header is received by mod_cache when requesting content by the backend it will attempt to handle it intelligently. If possible, mod_cache will detect the headers attributed in the "Vary" response in future requests and serve the correct cached response.

If for example, a response is received with a vary header such as;

```
Vary: negotiate, accept-language, accept-charset
```

mod_cache will only serve the cached content to requesters with accept-language and accept-charset headers matching those of the original request.
Authorization and Access Control

Using mod_cache is very much like having a built in reverse-proxy. Requests will be served by the caching module unless it determines that the backend should be queried. When caching local resources, this drastically changes the security model of Apache.

As traversing a filesystem hierarchy to examine potential .htaccess files would be a very expensive operation, partially defeating the point of caching (to speed up requests), mod_cache makes no decision about whether a cached entity is authorised for serving. In other words; if mod_cache has cached some content, it will be served from the cache as long as that content has not expired.

If, for example, your configuration permits access to a resource by IP address you should ensure that this content is not cached. You can do this by using the CacheDisable directive, or mod_expires. Left unchecked, mod_cache - very much like a reverse proxy - would cache the content when served and then serve it to any client, on any IP address.

Local exploits

As requests to end-users can be served from the cache, the cache itself can become a target for those wishing to deface or interfere with content. It is important to bear in mind that the cache must at all times be writable by the user which Apache is running as. This is in stark contrast to the usually recommended situation of maintaining all content unwritable by the Apache user.

If the Apache user is compromised, for example through a flaw in a CGI process, it is possible that the cache may be targeted.
When using `mod_disk_cache`, it is relatively easy to insert or modify a cached entity.

This presents a somewhat elevated risk in comparison to the other types of attack it is possible to make as the Apache user. If you are using `mod_disk_cache` you should bear this in mind - ensure you upgrade Apache when security upgrades are announced and run CGI processes as a non-Apache user using `suEXEC` if possible.

**Cache Poisoning**

When running Apache as a caching proxy server, there is also the potential for so-called cache poisoning. Cache Poisoning is a broad term for attacks in which an attacker causes the proxy server to retrieve incorrect (and usually undesirable) content from the backend.

For example if the DNS servers used by your system running Apache are vulnerable to DNS cache poisoning, an attacker may be able to control where Apache connects to when requesting content from the origin server. Another example is so-called HTTP request-smuggling attacks.

This document is not the correct place for an in-depth discussion of HTTP request smuggling (instead, try your favourite search engine) however it is important to be aware that it is possible to make a series of requests, and to exploit a vulnerability on an origin webserver such that the attacker can entirely control the content retrieved by the proxy.
The act of opening a file can itself be a source of delay, particularly on network filesystems. By maintaining a cache of open file descriptors for commonly served files, Apache can avoid this delay. Currently Apache provides two different implementations of File-Handle Caching.

**CacheFile**

The most basic form of caching present in Apache is the file-handle caching provided by `mod_file_cache`. Rather than caching file-contents, this cache maintains a table of open file descriptors. Files to be cached in this manner are specified in the configuration file using the `CacheFile` directive.

The `CacheFile` directive instructs Apache to open the file when Apache is started and to re-use this file-handle for all subsequent access to this file.

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

If you intend to cache a large number of files in this manner, you must ensure that your operating system's limit for the number of open files is set appropriately.

Although using `CacheFile` does not cause the file-contents to be cached per-se, it does mean that if the file changes while Apache is running these changes will not be picked up. The file will be consistently served as it was when Apache was started.
If the file is removed while Apache is running, Apache will continue to maintain an open file descriptor and serve the file as it was when Apache was started. This usually also means that although the file will have been deleted, and not show up on the filesystem, extra free space will not be recovered until Apache is stopped and the file descriptor closed.

**CacheEnable fd**

`mod_mem_cache` also provides its own file-handle caching scheme, which can be enabled via the `CacheEnable` directive.

```
CacheEnable fd /
```

As with all of `mod_cache` this type of file-handle caching is intelligent, and handles will not be maintained beyond the expiry time of the cached content.
In-Memory Caching

<table>
<thead>
<tr>
<th>Related Modules</th>
<th>Related Directives</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_mem_cache</td>
<td>CacheEnable</td>
</tr>
<tr>
<td>mod_file_cache</td>
<td>CacheDisable</td>
</tr>
<tr>
<td></td>
<td>MMapFile</td>
</tr>
</tbody>
</table>

Serving directly from system memory is universally the fastest method of serving content. Reading files from a disk controller or, even worse, from a remote network is orders of magnitude slower. Disk controllers usually involve physical processes, and network access is limited by your available bandwidth. Memory access on the other hand can take mere nano-seconds.

System memory isn't cheap though, byte for byte it's by far the most expensive type of storage and it's important to ensure that it is used efficiently. By caching files in memory you decrease the amount of memory available on the system. As we'll see, in the case of operating system caching, this is not so much of an issue, but when using Apache's own in-memory caching it is important to make sure that you do not allocate too much memory to a cache. Otherwise the system will be forced to swap out memory, which will likely degrade performance.

**Operating System Caching**

Almost all modern operating systems cache file-data in memory managed directly by the kernel. This is a powerful feature, and for the most part operating systems get it right. For example, on Linux, let’s look at the difference in the time it takes to read a file for the first time and the second time;

```
colg@coroebus:~$ time cat testfile > /dev/null
real    0m0.065s
user    0m0.000s
sys     0m0.001s
colm@coroebus:~$ time cat testfile > /dev/null
```
Even for this small file, there is a huge difference in the amount of time it takes to read the file. This is because the kernel has cached the file contents in memory.

By ensuring there is "spare" memory on your system, you can ensure that more and more file-contents will be stored in this cache. This can be a very efficient means of in-memory caching, and involves no extra configuration of Apache at all.

Additionally, because the operating system knows when files are deleted or modified, it can automatically remove file contents from the cache when necessary. This is a big advantage over Apache's in-memory caching which has no way of knowing when a file has changed.

Despite the performance and advantages of automatic operating system caching there are some circumstances in which in-memory caching may be better performed by Apache.

Firstly, an operating system can only cache files it knows about. If you are running Apache as a proxy server, the files you are caching are not locally stored but remotely served. If you still want the unbeatable speed of in-memory caching, Apache's own memory caching is needed.

**MMapFile Caching**

`mod_file_cache` provides the `MMapFile` directive, which allows you to have Apache map a static file's contents into memory at start time (using the mmap system call). Apache will use the in-memory contents for all subsequent accesses to this file.
As with the CacheFile directive, any changes in these files will not be picked up by Apache after it has started.

The MMapFile directive does not keep track of how much memory it allocates, so you must ensure not to over-use the directive. Each Apache child process will replicate this memory, so it is critically important to ensure that the files mapped are not so large as to cause the system to swap memory.

**mod_mem_cache Caching**

mod_mem_cache provides a HTTP-aware intelligent in-memory cache. It also uses heap memory directly, which means that even if MMap is not supported on your system, mod_mem_cache may still be able to perform caching.

Caching of this type is enabled via;

```
# Enable memory caching
CacheEnable mem /

# Limit the size of the cache to 1 Megabyte
MCacheSize 1024
```
mod_disk_cache provides a disk-based caching mechanism for mod_cache. As with mod_mem_cache this cache is intelligent and content will be served from the cache only as long as it is considered valid.

Typically the module will be configured as so;

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

Importantly, as the cached files are locally stored, operating system in-memory caching will typically be applied to their access also. So although the files are stored on disk, if they are frequently accessed it is likely the operating system will ensure that they are actually served from memory.

**Understanding the Cache-Store**

To store items in the cache, mod_disk_cache creates a 22 character hash of the URL being requested. This hash incorporates the hostname, protocol, port, path and any CGI arguments to the URL, to ensure that multiple URLs do not collide.

Each character may be any one of 64-different characters, which mean that overall there are $64^{22}$ possible hashes. For example, a URL might be hashed to xyTGxSM02b68mBCykqkp1w. This hash is used as a prefix for the naming of the files specific to that URL within the cache, however first it is split up into directories as
per the `CacheDirLevels` and `CacheDirLength` directives.

`CacheDirLevels` specifies how many levels of subdirectory there should be, and `CacheDirLength` specifies how many characters should be in each directory. With the example settings given above, the hash would be turned into a filename prefix as `/var/cache/apache/x/y/TGxSMO2b68mBCykqkp1w`.

The overall aim of this technique is to reduce the number of subdirectories or files that may be in a particular directory, as most file-systems slow down as this number increases. With setting of "1" for `CacheDirLength` there can at most be 64 subdirectories at any particular level. With a setting of 2 there can be 64 * 64 subdirectories, and so on. Unless you have a good reason not to, using a setting of "1" for `CacheDirLength` is recommended.

Setting `CacheDirLevels` depends on how many files you anticipate to store in the cache. With the setting of "2" used in the above example, a grand total of 4096 subdirectories can ultimately be created. With 1 million files cached, this works out at roughly 245 cached URLs per directory.

Each URL uses at least two files in the cache-store. Typically there is a ".header" file, which includes meta-information about the URL, such as when it is due to expire and a ".data" file which is a verbatim copy of the content to be served.

In the case of a content negotiated via the "Vary" header, a ".vary" directory will be created for the URL in question. This directory will have multiple ".data" files corresponding to the differently negotiated content.

**Maintaining the Disk Cache**

Although `mod_disk_cache` will remove cached content as it is
expired, it does not maintain any information on the total size of the cache or how little free space may be left.

Instead, provided with Apache is the `htcacheclean` tool which, as the name suggests, allows you to clean the cache periodically. Determining how frequently to run `htcacheclean` and what target size to use for the cache is somewhat complex and trial and error may be needed to select optimal values.

`htcacheclean` has two modes of operation. It can be run as persistent daemon, or periodically from cron. `htcacheclean` can take up to an hour or more to process very large (tens of gigabytes) caches and if you are running it from cron it is recommended that you determine how long a typical run takes, to avoid running more than one instance at a time.

![Figure 1: Typical cache growth / clean sequence.](image)

Because `mod_disk_cache` does not itself pay attention to how
much space is used you should ensure that `htcacheclean` is configured to leave enough "grow room" following a clean.
core
<table>
<thead>
<tr>
<th>ServerName</th>
<th>ServerAdmin</th>
<th>ServerSignature</th>
<th>ServerTokens</th>
<th>UseCanonicalName</th>
<th>UseCanonicalPhysicalPort</th>
</tr>
</thead>
</table>

- ServerAdmin, ServerTokens, UseCanonicalPhysicalPort

- ServerName: UseCanonicalName, URL
  - Apache
CoreDumpDirectory
DocumentRoot
ErrorLog
LockFile
PidFile
ScoreBoardFile
ServerRoot

Apache
<table>
<thead>
<tr>
<th>LimitRequestBody</th>
<th>LimitRequestFields</th>
<th>LimitRequestFieldsize</th>
<th>LimitRequestLine</th>
<th>RLimitCPU</th>
<th>RLimitMEM</th>
<th>RLimitNPROC</th>
<th>ThreadStackSize</th>
</tr>
</thead>
</table>

LimitRequest* Apache

RLimit* Apache fork

ThreadStackSize Netware
| Apache | id |
ErrorLog
LogLevel

ErrorLog

(unix
text) Unix

syslog

tail -f error_log

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

CGI
Apache httpd `mod_log_config`, `mod_log_agent` TransferLog

C `printf(1)`

Common Log Format

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

```
common
"\t"
```

CustomLog

Common Log Format (CLF)

```
```

```
127.0.0.1 (%h)
```
IdentityCheck On

frank (%u)
HTTP
401

:

[day/month/year:hour:minute:second zone]
day = 2*digit
month = 3*letter
year = 4*digit
hour = 2*digit
minute = 2*digit
second = 2*digit
zone = (`+' | `-' ) 4*digit

%{format}t

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

HTTP/1.0
"%r"

200 (%>s) (2)

2326 (%b)
Combined Log Format

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

Common Log Format

HTTP

"http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"

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

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

CustomLog
ReferLog
AgentLog

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

LogFormat
# Mark requests from the loop-back interface
SetEnvIf Remote_Addr "127\0\0\0\.1" dontlog
# Mark requests for the robots.txt file
SetEnvIf Request_URI "^/robots\.txt$" dontlog
# Log what remains
CustomLog logs/access_log common env=!dontlog

SetEnvIf Accept-Language "en" english
CustomLog logs/english_log common env=english
CustomLog logs/non_english_log common env=!english
mv access_log access_log.old
mv error_log error_log.old
apachectl graceful
sleep 600
gzip access_log.old error_log.old
Apache httpd

Apache httpd ID

Apache HTTP

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

<VirtualHost>

LogFormat "%v %l %u %t "%r" %>s %b" commonvhost
CustomLog logs/access_log commonvhost

%v
<table>
<thead>
<tr>
<th>Module</th>
<th>Configuration Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_logio</td>
<td>LogFormat</td>
</tr>
<tr>
<td></td>
<td>ForensicLog</td>
</tr>
<tr>
<td>mod_log_forensic</td>
<td>PidFile</td>
</tr>
<tr>
<td>mod_cgi</td>
<td>RewriteLog</td>
</tr>
<tr>
<td>mod_rewrite</td>
<td>RewriteLogLevel</td>
</tr>
<tr>
<td></td>
<td>ScriptLog</td>
</tr>
<tr>
<td></td>
<td>ScriptLogBuffer</td>
</tr>
<tr>
<td></td>
<td>ScriptLogLength</td>
</tr>
</tbody>
</table>

**mod_logio**

(log %I %O)

**Forensic**

**mod_log_forensic**

forensic

**PID**

Apache httpd ID

logs/httpd.pid

PidFile

ID

ScriptLog

CGI

**mod_rewrite**

RewriteLogLevel
URL

Apache URL
<table>
<thead>
<tr>
<th>Module</th>
<th>Configuration Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod.alias</td>
<td>Alias</td>
</tr>
<tr>
<td>mod.proxy</td>
<td>AliasMatch</td>
</tr>
<tr>
<td>mod.rewrite</td>
<td>CheckSpelling</td>
</tr>
<tr>
<td>mod.userdir</td>
<td>DocumentRoot</td>
</tr>
<tr>
<td>mod.speling</td>
<td>ErrorDocument</td>
</tr>
<tr>
<td>mod_vhost_alias</td>
<td>Options</td>
</tr>
<tr>
<td></td>
<td>ProxyPass</td>
</tr>
<tr>
<td></td>
<td>ProxyPassReverse</td>
</tr>
<tr>
<td></td>
<td>ProxyPassReverseCookieDomain</td>
</tr>
<tr>
<td></td>
<td>ProxyPassReverseCookiePath</td>
</tr>
<tr>
<td></td>
<td>Redirect</td>
</tr>
<tr>
<td></td>
<td>RedirectMatch</td>
</tr>
<tr>
<td></td>
<td>RewriteCond</td>
</tr>
<tr>
<td></td>
<td>RewriteMatch</td>
</tr>
<tr>
<td></td>
<td>ScriptAlias</td>
</tr>
<tr>
<td></td>
<td>ScriptAliasMatch</td>
</tr>
<tr>
<td></td>
<td>UserDir</td>
</tr>
</tbody>
</table>
Apache URL-Path

DocumentRoot

Apache

IP
DocumentRoot

DocumentRoot

SymLinksIfOwnerMatch

Alias /docs /var/web


ScriptAlias

AliasMatch

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

http://example.com/~user/cgi-bin/script.cgi /home/user/cgi-bin/script.cgi CGI
Unix  user  ~user/  mod_userdir

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

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

Userdir  /etc/passwd

"~"  (  %7e  )

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

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

DocumentRoot /foo/  /bar/

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

/foo/ URL-Path

www.example.com  /bar

Apache

RedirectMatch

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

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

/foo/ internal.example.com /bar/

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

ProxyPass internal.example.com
ProxyPassReverse internal.example.com
ProxyPassReverseCookieDomain internal.example.com
ProxyPassReverseCookiePath Cookie

internal.example.com

internal.example.com mod_proxy_html

HTML XHTML
mod_rewrite

mod_rewrite
URL

"File Not Found"

mod_speling ()

URL

mod_speling
mod_speling

URL unix

ErrorDocument

Apache HTTP 404
Security Tips

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

Of course, most times that a web server is compromised, it is not because of problems in the HTTP Server code. Rather, it comes from problems in add-on code, CGI scripts, or the underlying Operating System. You must therefore stay aware of problems and updates with all the software on your system.
In typical operation, Apache is started by the root user, and it switches to the user defined by the `User` directive to serve hits. As is the case with any command that root executes, you must take care that it is protected from modification by non-root users. Not only must the files themselves be writeable only by root, but so must the directories, and parents of all directories. For example, if you choose to place ServerRoot in `/usr/local/apache` then it is suggested that you create that directory as root, with commands like these:

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

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

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

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

If you allow non-root users to modify any files that root either executes or writes on then you open your system to root compromises. For example, someone could replace the `httpd` binary so that the next time you start it, it will execute some arbitrary code. If the logs directory is writeable (by a non-root user), someone could replace a log file with a symlink to some
other system file, and then root might overwrite that file with arbitrary data. If the log files themselves are writeable (by a non-root user), then someone may be able to overwrite the log itself with bogus data.
Server Side Includes (SSI) present a server administrator with several potential security risks.

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

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

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

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

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

Another solution is to disable the ability to run scripts and programs from SSI pages. To do this replace Includes with IncludesNOEXEC in the Options directive. Note that users may still use <!--#include virtual="..." --> to execute CGI scripts if these scripts are in directories designated by a ScriptAlias directive.
First of all, you always have to remember that you must trust the writers of the CGI scripts/programs or your ability to spot potential security holes in CGI, whether they were deliberate or accidental. CGI scripts can run essentially arbitrary commands on your system with the permissions of the web server user and can therefore be extremely dangerous if they are not carefully checked.

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

- You trust your users not to write scripts which will deliberately or accidentally expose your system to an attack.
- You consider security at your site to be so feeble in other areas, as to make one more potential hole irrelevant.
- You have no users, and nobody ever visits your server.
Limiting CGI to special directories gives the admin control over what goes into those directories. This is inevitably more secure than non script aliased CGI, but only if users with write access to the directories are trusted or the admin is willing to test each new CGI script/program for potential security holes.

Most sites choose this option over the non script aliased CGI approach.
Embedded scripting options which run as part of the server itself, such as mod_php, mod_perl, mod_tcl, and mod_python, run under the identity of the server itself (see the User directive), and therefore scripts executed by these engines potentially can access anything the server user can. Some scripting engines may provide restrictions, but it is better to be safe and assume not.
To run a really tight ship, you'll want to stop users from setting up .htaccess files which can override security features you've configured. Here's one way to do it.

In the server configuration file, put

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

This prevents the use of .htaccess files in all directories apart from those specifically enabled.
One aspect of Apache which is occasionally misunderstood is the feature of default access. That is, unless you take steps to change it, if the server can find its way to a file through normal URL mapping rules, it can serve it to clients.

For instance, consider the following example:

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

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

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

This will forbid default access to filesystem locations. Add appropriate `Directory` blocks to allow access only in those areas you wish. For example,

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

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

Also be wary of playing games with the `UserDir` directive; setting
it to something like ./ would have the same effect, for root, as the first example above. If you are using Apache 1.3 or above, we strongly recommend that you include the following line in your server configuration files:

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

A couple of examples:

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

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

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

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

```
./.htpasswd HTTP/1.1"
```

in your Access Log. This means you probably commented out the following in your server configuration file:

```
<Files ~ "^\.ht">
Order allow,deny
Deny from all
</Files>
```
Apache HTTP
Shared Object) (DSO)  DSO

DSO
Apache 2.2 DSO

1. Apache mod_foo.c DSO mod_foo.so:

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

2. Apache mod_foo.c DSO mod_foo.so:

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

3. Apache:

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

4. Apache mod_foo.c apxs Apache:

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

   httpd.conf  Loa
Unix OS (DSO) /

: 1

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

DSO (DSO )
)

DSO DSO (dlopen() DSO )
)

DSO API

DSO : DSO

DSO

1998 DSO : Perl 5
Apache
Apache DSO
DSO:

- configure Apache (SSL [mod_perl, PHP3]) Apache
- Apache DSO apxs Apache
  `apxs -i apachectl restart`

DSO:

- Unix 20%
- (PIC) (position independent code)

- DSO DSO (DS)
  DSO Apache PI
dlopen ()
Apache HTTP/1.1

mod_negotiation
Accept-Language: fr

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

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

URI (RFC 2396) Apache
0 1

HTTP
variant

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

**type-map**

*type-map* (Apache *MIME* type-map)

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

Variant

```
) foo foo.
```

<table>
<thead>
<tr>
<th>URI</th>
<th>Content-type</th>
<th>Content-language</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>text/html</td>
<td></td>
</tr>
<tr>
<td>foo.en.html</td>
<td>text/html</td>
<td>en</td>
</tr>
<tr>
<td>foo.fr.de.html</td>
<td>text/html;charset=iso-8859-2</td>
<td>fr, de</td>
</tr>
</tbody>
</table>

**MultiViews**

"qs"

<table>
<thead>
<tr>
<th>URI</th>
<th>Content-type</th>
<th>qs</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>image/jpeg</td>
<td>0.8</td>
</tr>
<tr>
<td>foo.gif</td>
<td>image/gif</td>
<td>0.5</td>
</tr>
<tr>
<td>foo.txt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
qs 0.000 1.000 qs
variant qs 1.0 qs
JPEG ASCII
qs variant

mod_negotiation

**Multiviews**

```
<Files>
  AllowOverride All MultiViews
</Files>
```

Multiviews: /some/dir/foo

```
<Directory>
  MultiViews /some/dir/foo
  DirectoryIndex index

  index.html index.html3

  MultiViews
```

```
Apache variant

1. **Apache Server** driven negotiation Apa
   Apache

2. RFC 2295
   Variant
   2296 'remote variant selection algorithm'

```
Accept
Accept-Language
Accept-Encoding
Accept-Charset

Apache
variant ()

1. Accept
2. variant
3. ()
   Accept-Language
   LanguagePriority

```
4. (text/html )

5. Accept-Charset  
   text/*

6. ISO-8859-1  
   variant

7. variant  
   user-agent  
   variant

8. variant

9. variant

3. variant

4. variant  
   ()  
   representation" )  
   variant HTML  
   Vary
Accept:

Accept: image/*, */*

"image/*"

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

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

1.0 ()

Accept: q
"type/*" 0.02 q

Apache "*/*" 0.01 q

Apache 2.0

"Multiple Choices"

LanguagePriority
Language en-GB en
Acceptable Variants
LanguagePriority
en Apache
"fr" "fr"

(Cookie URL)

mod_negotiation prefer-language
mod_negotiation variant

Example

SetEnvIf Cookie "language=(.+)" prefer-language=$1
Apache transparent content negotiation (RFC 2295) variant {encoding ..} variant variant
Accept-Encoding variant variant RVSA/1.0
(RFC 2296) RVSA/1.0 variant 5
MIME (html) (gz):

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

<table>
<thead>
<tr>
<th>foo.html.en</th>
<th>foo</th>
<th>foo.html</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo.en.html</td>
<td>foo</td>
<td>foo.html</td>
<td>foo.html</td>
</tr>
<tr>
<td>foo.html.gz</td>
<td>foo</td>
<td>foo.html.gz</td>
<td></td>
</tr>
<tr>
<td>foo.en.html.gz</td>
<td>foo</td>
<td>foo.html.gz</td>
<td></td>
</tr>
<tr>
<td>foo.gz.html.en</td>
<td>foo gz</td>
<td>foo.html.gz</td>
<td></td>
</tr>
<tr>
<td>foo.html.gz.en</td>
<td>foo html</td>
<td>foo.gz</td>
<td></td>
</tr>
</tbody>
</table>

( foo)

MIME ( foo.html) ()
URL
HTTP/1.0
HTTP/1.0
HTTP/1.1
HTTP/1.1
Apache

"500 Server Error"
NCSA httpd 1.3 /

1. NCSA
2. URL
3. URL

URL /

Apache CGI:

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

REDIRECT_
REDIRECT_URL  REDIRECT_QUERY_STRING  URL(CGI CGI)

"
AllowOverride .htaccess ErrorDocument

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

ErrorDocument <3-digit-code> <action>

action ()

1. ("")
2. URL
3. URL
... 
print "Content-type: text/html\n";
printf "Status: %s Condition Intercepted\n",
ENV{"REDIRECT_STATUS"};
...

404 Not Found
Apache

DNS
<VirtualHost

Apache
IP Apache
Listen

Listen
Listen
listen

80 8000

Listen 80
Listen 8000

Listen 192.0.2.1:80
Listen 192.0.2.5:8000

IPv6

Listen [2001:db8::a00:20ff:fea7:ccea]:80
<table>
<thead>
<tr>
<th>APR</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6</td>
<td>IPv4</td>
</tr>
</tbody>
</table>

IPv6

**configure**  Apache

Linux  Tru64  IPv4  IPv6
IPv6  --enable-v4-mapped  **configure**

**configure**  FreeBSD, NetBSD, OpenBSD

Apache

<table>
<thead>
<tr>
<th>APR</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>IPv6</td>
</tr>
</tbody>
</table>

components

Listen 0.0.0.0:80
Listen 192.0.2.1:80

Apache  IPv4  IPv6
mapped  **configure**  FreeBSD, NetBSD, OpenBSD
Listen

<VirtualHost>

<VirtualHost>

<VirtualHost>

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0.
Apache HTTP
Apache HTTP

Apache 2.0

- Apache
  - Apache 1.3 POSIX
- MPM Apache

MPM MPM
configure --with-mpm=NAME

./httpd -l
<table>
<thead>
<tr>
<th>OS</th>
<th>MPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeOS</td>
<td>beos</td>
</tr>
<tr>
<td>Netware</td>
<td>mpm_netware</td>
</tr>
<tr>
<td>OS/2</td>
<td>mpmt_os2</td>
</tr>
<tr>
<td>Unix</td>
<td>prefork</td>
</tr>
<tr>
<td>Windows</td>
<td>mpm_winnt</td>
</tr>
</tbody>
</table>
Apache

Apache HTTP
<table>
<thead>
<tr>
<th>mod_env</th>
<th>BrowserMatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_rewrite</td>
<td>BrowserMatchNoCase</td>
</tr>
<tr>
<td>mod_setenvif</td>
<td>PassEnv</td>
</tr>
<tr>
<td>mod_unique_id</td>
<td>RewriteRule</td>
</tr>
<tr>
<td></td>
<td>SetEnv</td>
</tr>
<tr>
<td></td>
<td>SetEnvIf</td>
</tr>
<tr>
<td></td>
<td>SetEnvIfNoCase</td>
</tr>
<tr>
<td></td>
<td>UnsetEnv</td>
</tr>
</tbody>
</table>

Apache

mod_setenvif

referrer HTTP Referer

RewriteRule

mod_unique_id

**CGI**

Apache

<table>
<thead>
<tr>
<th>CGI</th>
<th>CGI</th>
<th>suexexec</th>
<th>CGI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(:'_)
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_authz_host</td>
<td>Allow</td>
</tr>
<tr>
<td>mod_cgi</td>
<td>CustomLog</td>
</tr>
<tr>
<td>mod_ext_filter</td>
<td>Deny</td>
</tr>
<tr>
<td>mod_headers</td>
<td>ExtFilterDefine</td>
</tr>
<tr>
<td>mod_include</td>
<td>Header</td>
</tr>
<tr>
<td>mod_log_config</td>
<td>LogFormat</td>
</tr>
<tr>
<td>mod_rewrite</td>
<td>RewriteCond</td>
</tr>
<tr>
<td></td>
<td>RewriteRule</td>
</tr>
</tbody>
</table>

**CGI**

**SSI**

| mod_include | INCLUDES | server-parsed (SSI) |

allow from env= deny from env=

LogFormat

gif

Header
ExtFilterDefine
enableenv=

URL
RewriteCond %{ENV:...} mod_rewrite
  ENV:
PassEnv

downgrade-1.0
HTTP/1.0 HTTP/1.0

force-gzip
DEFLATE
accept-encoding

force-no-vary
Vary

force-response-1.0
HTTP/1.0
HTTP/1.1

gzip-only-text/html
1 text/html
(mod_deflate
(gzip "identity")

no-gzip
(mod_deflate DEFLATE

nokeepalive
KeepAlive

prefer-language
(mod_negotiation (en, ja, x-klingon)

variant
redirect-carefully

suppress-error-charset

*Apache 2.2*

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

mod_proxy    mod_proxy
The following directives modify normal HTTP response behavior. The first directive disables keepalive for Netscape 2.x and browsers that spoof it. There are known problems with these browser implementations. The second directive is for Microsoft Internet Explorer 4.0b2 which has a broken HTTP/1.1 implementation and does not properly support keepalive when it is used on 301 or 302 (redirect) responses.

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\0b2;" nokeepalive downgrade-1.0 force-response-1.0

The following directive disables HTTP/1.1 responses to browsers which are in violation of the HTTP/1.0 spec by not being able to grok a basic 1.1 response.

BrowserMatch "RealPlayer 4\0" force-response-1.0
BrowserMatch "Java/1\0.0" force-response-1.0
BrowserMatch "JDK/1\0.0" 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

SetEnvIf Referer "^http://www.example.com/" local_referal
# Allow browsers that do not send Referer info
SetEnvIf Referer "^$" local_referal
<Directory /web/images>
    Order Deny,Allow
    Deny from all
    Allow from env=local_referal
</Directory>

ApacheToday  Keeping Your Images from Adorning Other Sites
| mod_actions | Action          |
| mod_asis    | AddHandler     |
| mod_cgi     | RemoveHandler  |
| mod_imagemap| SetHandler     |
| mod_info    |                |
| mod_mime    |                |
| mod_negotiation |          |
| mod_status  |                |

Apache

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

(mod_status)
CGI

html footer.pl CGI

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

CGI

HTTP

send-as-is HTTP
send-as-is

.splitContainer

<Directory /web/htdocs/asis>
SetHandler send-as-is
</Directory>
char *handler

( : "-" )
Apache
Apache 2.0

Request Processing in Apache
Apache:

- mod_include SSI
- mod_ssl SSL (https)
- mod_deflate on-the-fly
- mod_charset_lite
- mod_ext_filter

Apache modules.apache.org:

- HTML
- XML
- XSLT
- XIncludes
- XML
- HTML
- PHP
Apache 2.1  mod_filter  HTML  HTML  JPEG

- HTML  text/html  application/xhtml+xml
- 
- 
-
: ()

AddInputFilter, AddOutputFilter, RemoveInputFilter, RemoveOutputFilter

mod_filter, FilterDeclare, FilterProvider

AddOutputFilterByType
<table>
<thead>
<tr>
<th>suEXEC</th>
<th>Apache</th>
<th>Web</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSI</td>
<td>web</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>suEXEC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Apache

1  **setuid  setgid**  UNIX

2

3  suEXEC  suEXEC
Apache  suEXEC

4  suEXEC  Apache
suEXEC  suEXEC
suEXEC

? !
suEXEC

**suEXEC** Apache web

setuid "wrapper"

HTTP wrapper

Apache wrapper

wrapper

1. **wrapper ?

   wrapper

2. **wrapper ?

   wrapper

   Apache

   suEXEC

3. **wrapper ?

   wrapper ?

   (Apache )

4. **CGI, SSI ?

   CGI, SSI '/'

   ' . ' ?

   -with-suexec-docroot=DIR )

5. ?

   ?

6. ?

   ?

7. ?

   **suEXEC root CGI/SSI**
8. **ID** **ID**

    **ID**

8. **ID** **CGI/SSI** **ID**

9. 

    **suEXEC** 'root' **CGI/SSI**

10. **ID** **ID**

    **ID**

11. **wrapper**

    **setuid** **setgid**

12. **CGI/SSI**

    (change directory)

13. **Apache**

    **suEXEC**

    (suEXEC)

14. 

15. **CGI/SSI**

16. **CGI/SSI**

    **CGI/SSI**

17. **CGI/SSI** **setuid** **setgid**
UID/GID

18. I

19. ?

suEXEC

20. CGI/SSI exec?

suEXEC

suEXEC wrapper
suEXEC

--enable-suexec
 enable-suexec

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

--with-suexec-caller=UID
 Apache suexec

--with-suexec-userdir=DIR
 suEXEC Userdir
 ("*"

 UserDir

--with-suexec-docroot=DIR
 Apache suEXEC (UserDir
 "/htdocs"
 wrapper "/home/apache/htdocs"

--with-suexec-uidmin=UID
 suEXEC UID 500 10

--with-suexec-gidmin=GID
 suEXEC GID 100

--with-suexec-logfile=FILE
 suEXEC logfiledir

--with-suexec-safepath=PATH
 CGI PATH "/usr/local/bin:

suEXEC wrapper
--enable-suexec suEXEC
(Apache)

"make"

make install

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

root wrapper ID

suEXEC
--with-suexec-caller configure

suEXEC

User www
Group webgroup

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

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

Apache suEXEC
suEXEC

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

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

wrapper

suEXEC Apache

suEXEC suexec Apache kill
suEXEC wrapper

mod_userdir

--with-suexec-userdir
suEXEC wrapper --with-suexec-logfile
suEXEC

- suEXEC
- suEXEC PATH

suEXEC

Apache

suEXEC

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

Compared to Apache 1.3, release 2.x contains many additional optimizations to increase throughput and scalability. Most of these improvements are enabled by default. However, there are compile-time and run-time configuration choices that can significantly affect performance. This document describes the options that a server administrator can configure to tune the performance of an Apache 2.x installation. Some of these configuration options enable the httpd to better take advantage of the capabilities of the hardware and OS, while others allow the administrator to trade functionality for speed.
The single biggest hardware issue affecting webserver performance is RAM. A webserver should never ever have to swap, as swapping increases the latency of each request beyond a point that users consider "fast enough". This causes users to hit stop and reload, further increasing the load. You can, and should, control the MaxClients setting so that your server does not spawn so many children that it starts swapping. The procedure for doing this is simple: determine the size of your average Apache process, by looking at your process list via a tool such as top, and divide this into your total available memory, leaving some room for other processes.

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

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

- Run the latest stable release and patch level of the operating system that you choose. Many OS suppliers have introduced significant performance improvements to their TCP stacks and thread libraries in recent years.

- If your OS supports a sendfile(2) system call, make sure you install the release and/or patches needed to enable it. (With Linux, for example, this means using Linux 2.4 or later. For early releases of Solaris 8, you may need to apply a patch.) On systems where it is available, sendfile enables Apache 2 to deliver static content faster and with lower CPU utilization.
### Related Modules
- mod_dir
- mpm_common
- mod_status

### Related Directives
- AllowOverride
- DirectoryIndex
- HostnameLookups
- EnableMMAP
- EnableSendfile
- KeepAliveTimeout
- MaxSpareServers
- MinSpareServers
- Options
- StartServers

---

**HostnameLookups and other DNS considerations**

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

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

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

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

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

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

**FollowSymLinks and SymLinksIfOwnerMatch**

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

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

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

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

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

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

**AllowOverride**

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

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

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

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

```bash
DirectoryIndex index
```

Use a complete list of options:

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

where you list the most common choice first.

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

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

**Memory-mapping**

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

On some platforms, this memory-mapping improves performance. However, there are cases where memory-mapping can hurt the performance or even the stability of the httpd:
• On some operating systems, mmap does not scale as well as read(2) when the number of CPUs increases. On multiprocessor Solaris servers, for example, Apache 2.x sometimes delivers server-parsed files faster when mmap is disabled.

• If you memory-map a file located on an NFS-mounted filesystem and a process on another NFS client machine deletes or truncates the file, your process may get a bus error the next time it tries to access the mapped file content.

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

**Sendfile**

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

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

• Some platforms may have broken sendfile support that the build system did not detect, especially if the binaries were built on another box and moved to such a machine with broken sendfile support.

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

For installations where either of these factors applies, you should
use EnableSendfile off to disable sendfile delivery of file contents. (Note: This directive can be overridden on a per-directory basis.)

**Process Creation**

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

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

This appears to be responsive enough that it's almost unnecessary to twiddle the `MinSpareServers`, `MaxSpareServers` and `StartServers` knobs. When more than 4 children are spawned per second, a message will be emitted to the `ErrorLog`. If you see a lot of these errors, then consider
tuning these settings. Use the `mod_status` output as a guide.

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

When keep-alives are in use, children will be kept busy doing nothing waiting for more requests on the already open connection. The default `KeepAliveTimeout` of 5 seconds attempts to minimize this effect. The tradeoff here is between network bandwidth and server resources. In no event should you raise this above about 60 seconds, as `most of the benefits are lost`. 
Choosing an MPM

Apache 2.x supports pluggable concurrency models, called Multi-Processing Modules (MPMs). When building Apache, you must choose an MPM to use. There are platform-specific MPMs for some platforms: beos, mpm_netware, mpmt_os2, and mpm_winnt. For general Unix-type systems, there are several MPMs from which to choose. The choice of MPM can affect the speed and scalability of the httpd:

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

For more information on these and other MPMs, please see the MPM documentation.

Modules

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

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

An associated question that arises here is, of course, what modules you need, and which ones you don't. The answer here will, of course, vary from one web site to another. However, the minimal list of modules which you can get by with tends to include mod_mime, mod_dir, and mod_log_config. mod_log_config is, of course, optional, as you can run a web site without log files. This is, however, not recommended.

**Atomic Operations**

Some modules, such as mod_cache and recent development builds of the worker MPM, use APR's atomic API. This API provides atomic operations that can be used for lightweight thread synchronization.

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

```
./buildconf
./configure --with-mpm=worker --enable-nonportable-atomics=yes
```
The --enable-nonportable-atomics option is relevant for the following platforms:

- Solaris on SPARC
  By default, APR uses mutex-based atomics on Solaris/SPARC. If you configure with --enable-nonportable-atomics, however, APR generates code that uses a SPARC v8plus opcode for fast hardware compare-and-swap. If you configure Apache with this option, the atomic operations will be more efficient (allowing for lower CPU utilization and higher concurrency), but the resulting executable will run only on UltraSPARC chips.

- Linux on x86
  By default, APR uses mutex-based atomics on Linux. If you configure with --enable-nonportable-atomics, however, APR generates code that uses a 486 opcode for fast hardware compare-and-swap. This will result in more efficient atomic operations, but the resulting executable will run only on 486 and later chips (and not on 386).

**mod_status and ExtendedStatus On**

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

**accept Serialization - Multiple Sockets**

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

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

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

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

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

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

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

```c
for (;;) {
  accept_mutex_on ();
  for (;;) {
    fd_set accept_fds;
    FD_ZERO (&accept_fds);
    for (i = first_socket; i <= last_socket; ++i) {
```
The functions accept_mutex_on and accept_mutex_off implement a mutual exclusion semaphore. Only one child can have the mutex at any time. There are several choices for implementing these mutexes. The choice is defined in src/conf.h (pre-1.3) or src/include/ap_config.h (1.3 or later). Some architectures do not have any locking choice made, on these architectures it is unsafe to use multiple Listen directives.

The directive AcceptMutex can be used to change the selected mutex implementation at run-time.

**AcceptMutex flock**
This method uses the flock(2) system call to lock a lock file (located by the LockFile directive).

**AcceptMutex fcntl**
This method uses the fcntl(2) system call to lock a lock file (located by the LockFile directive).

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

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

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

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

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

Ideally you should run servers without multiple `listen` statements if you want the highest performance. But read on.

**accept Serialization - Single Socket**

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

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

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

When this feature was added to Apache, it caused a flurry of problems on various versions of Unix because of shortsightedness. The TCP specification does not state that the FIN_WAIT_2 state has a timeout, but it doesn't prohibit it. On systems without the timeout, Apache 1.2 induces many sockets stuck forever in the FIN_WAIT_2 state. In many cases this can be avoided by simply upgrading to the latest TCP/IP patches supplied by the vendor. In cases where the vendor has never released patches (*i.e.*, SunOS4 -- although folks with a source license can patch it themselves), we have decided to disable this feature.

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

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

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

    /* shutdown the sending side */
    shutdown (s, 1);
}```
This naturally adds some expense at the end of a connection, but it is required for a reliable implementation. As HTTP/1.1 becomes more prevalent, and all connections are persistent, this expense will be amortized over more requests. If you want to play with fire and disable this feature, you can define NO_LINGCLOSE, but this is not recommended at all. In particular, as HTTP/1.1 pipelined persistent connections come into use, lingering_close is an absolute necessity (and pipelined connections are faster, so you want to support them).

**Scoreboard File**

Apache's parent and children communicate with each other through something called the scoreboard. Ideally this should be implemented in shared memory. For those operating systems that we either have access to, or have been given detailed ports for, it typically is implemented using shared memory. The rest default to using an on-disk file. The on-disk file is not only slow, but it is unreliable (and less featured). Peruse the src/main/conf.h file for your architecture, and look for either USE_MMAP_SCOREBOARD or USE_SHMGET_SCOREBOARD. Defining one of those two (as well
as their companions HAVE_MMAP and HAVE_SHMGET respectively) enables the supplied shared memory code. If your system has another type of shared memory, edit the file src/main/http_main.c and add the hooks necessary to use it in Apache. (Send us back a patch too, please.)

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

**DYNAMIC_MODULE_LIMIT**

If you have no intention of using dynamically loaded modules (you probably don't if you're reading this and tuning your server for every last ounce of performance), then you should add `-DDYNAMIC_MODULE_LIMIT=0` when building your server. This will save RAM that's allocated only for supporting dynamically loaded modules.
Here is a system call trace of Apache 2.0.38 with the worker MPM on Solaris 8. This trace was collected using:

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

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

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

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

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

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

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

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

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

```
/65:  getsockname(9, 0x00200BA4, 0x00200BC4, 1) = 0
```
In order to implement virtual hosts, Apache needs to know the local socket address used to accept the connection. It is possible to eliminate this call in many situations (such as when there are no virtual hosts, or when *Listen* directives are used which do not have wildcard addresses). But no effort has yet been made to do these optimizations.

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

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

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

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

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

The worker thread reads the request from the client.

```plaintext
/65:    stat("/var/httpd/apache/httpd-8999/htdocs/10k.html", 0xFAF7B978) = 0
/65:    open("/var/httpd/apache/httpd-8999/htdocs/10k.html", 0_RDONLY) = 10
```
This httpd has been configured with Options FollowSymLinks and AllowOverride None. Thus it doesn't need to \texttt{lstat(2)} each directory in the path leading up to the requested file, nor check for .htaccess files. It simply calls \texttt{stat(2)} to verify that the file: 1) exists, and 2) is a regular file, not a directory.

\begin{verbatim} /65: sendfilev(0, 9, 0x00200F90, 2, 0xF47B53C) = 10269 \end{verbatim}

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

\begin{verbatim} /65: write(4, " 1 2 7 . 0 . 0 . 1 - ", 78) = 78 \end{verbatim}

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

\begin{verbatim} /65: shutdown(9, 1, 1) = 0 /65: poll(0xF47B980, 1, 2000) = 1 /65: read(9, 0xF47BC20, 512) = 0 /65: close(9) = 0 \end{verbatim}

The worker thread does a lingering close of the connection.

\begin{verbatim} /65: close(10) = 0 /65: lwp_park(0x00000000, 0) (sleeping...) \end{verbatim}

Finally the worker thread closes the file that it has just delivered
and blocks until the listener assigns it another connection.

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

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

DocumentRoot
NameVirtualHost
ServerAlias
ServerName
ServerPath
VirtualHost
<VirtualHost>

NameVirtualHost * IP
*:80 NameVirtualHost

<VirtualHost>
NameVirtualHost (IP

ServerName

ServerName DocumentRoot

www.domain.tld

httpd.conf

NameVirtualHost *:80

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

<VirtualHost *:80>
ServerPath

NameVirtualHost 111.22.33.44

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

"/domain" URI

http://www.domain.tld/domain/ Host

http://www.domain.tld/

http://www.domain.tld/domain/misc/file.html

http://www.domain.tld/domain/misc/image.gif

(: "http://www.domain.tld/domain/misc/file.html" "/domain/misc/file.html"
Apache HTTP 2.2
Apache IP
IP

"ifconfig")

▲
httpd

Listen www.smallco.com:80

IP
**httpd**

**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>
```

**suEXEC**

```
SuexecUserGroup VirtualHost
```

: Apache
Dynamically configured mass virtual hosting

This document describes how to efficiently serve an arbitrary number of virtual hosts with Apache.
The techniques described here are of interest if your `httpd.conf` contains many `<VirtualHost>` sections that are substantially the same, for example:

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

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

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

The main disadvantage is that you cannot have a different log file for each virtual host; however if you have very many virtual hosts then doing this is dubious anyway because it eats file descriptors. It is better to log to a pipe or a fifo and arrange for the process at the other end to distribute the logs to the customers (it can also accumulate statistics, etc.).
A virtual host is defined by two pieces of information: its IP address, and the contents of the Host: header in the HTTP request. The dynamic mass virtual hosting technique is based on automatically inserting this information into the pathname of the file that is used to satisfy the request. This is done most easily using mod_vhost_alias, but if you are using a version of Apache up to 1.3.6 then you must use mod_rewrite. Both of these modules are disabled by default; you must enable one of them when configuring and building Apache if you want to use this technique.

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

The other thing to `fake' is the document root (configured with DocumentRoot and available to CGIs via the DOCUMENT_ROOT environment variable). In a normal configuration this setting is used by the core module when mapping URIs to filenames, but when the server is configured to do dynamic virtual hosting that job is taken over by another module (either mod_vhost_alias or mod_rewrite) which has a different way of doing the mapping.
Neither of these modules is responsible for setting the DOCUMENT_ROOT environment variable so if any CGIs or SSI documents make use of it they will get a misleading value.
This extract from httpd.conf implements the virtual host arrangement outlined in the Motivation section above, but in a generic fashion using mod_vhost_alias.

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

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

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

This configuration can be changed into an IP-based virtual hosting solution by just turning UseCanonicalName Off into UseCanonicalName DNS. The server name that is inserted into the filename is then derived from the IP address of the virtual host.
This is an adjustment of the above system tailored for an ISP's homepages server. Using a slightly more complicated configuration we can select substrings of the server name to use in the filename so that e.g. the documents for www.user.isp.com are found in /home/user/. It uses a single cgi-bin directory instead of one per virtual host.

```
# all the preliminary stuff is the same as above, then

# include part of the server name in the filenames
VirtualDocumentRoot /www/hosts/%2/docs

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

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

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

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

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

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

  CustomLog logs/access_log.commercial vcommon

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

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

  CustomLog logs/access_log家用pages vcommon

  VirtualDocumentRoot /www/homepages/%0/docs
  ScriptAlias /cgi-bin/ /www/std-cgi/
</VirtualHost>
```
If the first VirtualHost block does not include a `ServerName` directive, the reverse DNS of the relevant IP will be used instead. If this is not the server name you wish to use, a bogus entry (`ServerName none.example.com`) can be added to get around this behaviour.
After the first example I noted that it is easy to turn it into an IP-based virtual hosting setup. Unfortunately that configuration is not very efficient because it requires a DNS lookup for every request. This can be avoided by laying out the filesystem according to the IP addresses themselves rather than the corresponding names and changing the logging similarly. Apache will then usually not need to work out the server name and so incur a DNS lookup.

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

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

# include the IP address in the filenames
VirtualDocumentRootIP /www/hosts/%0/docs
VirtualScriptAliasIP /www/hosts/%0/cgi-bin
This extract from httpd.conf does the same thing as the first example. The first half is very similar to the corresponding part above but with some changes for backward compatibility and to make the mod_rewrite part work properly; the second half configures mod_rewrite to do the actual work.

There are a couple of especially tricky bits: By default, mod_rewrite runs before the other URI translation modules (mod_alias etc.) so if they are used then mod_rewrite must be configured to accommodate them. Also, some magic must be performed to do a per-dynamic-virtual-host equivalent of ScriptAlias.

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

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

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

# now for the hard bit
RewriteEngine On

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

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

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

# that's it!
This does the same thing as the second example.

```
RewriteEngine on

RewriteMap lowercase int:tolower

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

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

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

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

# define the global CGI directory
ScriptAlias /cgi-bin/ /www/std-cgi/
```
This arrangement uses more advanced mod_rewrite features to get the translation from virtual host to document root from a separate configuration file. This provides more flexibility but requires more complicated configuration.

The vhost.map file contains something like this:

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

The http.conf contains this:

```
RewriteEngine on

RewriteMap lowercase int:tolower

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

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

RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond %{lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond %{vhost:%1} ^(/.*)$
RewriteRule ^/(.*)$ %1/cgi-bin/$1
```
# Ensure that Apache listens on port 80
Listen 80

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

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

    # Other directives here
</VirtualHost>

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

    # Other directives here
</VirtualHost>

NameVirtualHost 172.20.30.40
<VirtualHost 172.20.30.40>
# etc ...

IP

IP
Listen 80

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

# This is the other address
NameVirtualHost 172.20.30.50

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

  # Other directives here ...
</VirtualHost>

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

  # Other directives here ...
</VirtualHost>
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

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

VirtualHost
<table>
<thead>
<tr>
<th>IP</th>
<th>&quot;NameVirtualHost&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name:port</td>
<td>Listen</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;VirtualHost name:port&gt;</td>
<td></td>
</tr>
</tbody>
</table>

| Listen 80   | NameVirtualHost 172.20.30.40:80 | NameVirtualHost 172.20.30.40:8080 |
| Listen 8080 | <VirtualHost 172.20.30.40:80>  | ServerName www.example.com          |
|             | DocumentRoot /www/domain-80     |   |

| <VirtualHost 172.20.30.40:8000> | ServerName www.example.com | DocumentRoot /www/domain-8080 |
| <VirtualHost 172.20.30.40:80>  | ServerName www.example.org  | DocumentRoot /www/otherdomain-80 |

| <VirtualHost 172.20.30.40:8080> | ServerName www.example.org  | DocumentRoot /www/otherdomain-8080 |
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)
</VirtualHost>
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>
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>
<VirtualHost *:*>
ProxyPreserveHost On
ProxyPass / http://192.168.111.2
ProxyPassReverse / http://192.168.111.2/
ServerName hostname.example.com
</VirtualHost>
```xml
<VirtualHost _default_:80>
    DocumentRoot /www/default80
    # ...
</VirtualHost>

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

80 ( 

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

80 ( 

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

80
<VirtualHost _default_:80>
DocumentRoot /www/default
...
</VirtualHost>
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>
NameVirtualHost 172.20.30.40

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

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

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

**ServerPath** | **URL**
--- | ---
http://www.sub1.domain.tld/sub1/ | sub1-vhost
URL http://www.sub1.domain.tld/ | vhost
Host: | Host:

sub1-vhost

Host: | URL
--- | ---
http://www.sub2
An In-Depth Discussion of Virtual Host Matching

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

If you just want to make it work without understanding how, here are some examples.
There is a *main_server* which consists of all the definitions appearing outside of `<VirtualHost>` sections. There are virtual servers, called *vhosts*, which are defined by `<VirtualHost>` sections.

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

The main_server has no default ServerPath, or ServerAlias. The default ServerName is deduced from the server's IP address.

Port numbers specified in the `VirtualHost` directive do not influence what port numbers Apache will listen on, they only discriminate between which `VirtualHost` will be selected to handle a request.

Each address appearing in the `VirtualHost` directive can have an optional port. If the port is unspecified it is treated as a wildcard port. The special port `*` indicates a wildcard that matches any port. Collectively the entire set of addresses (including multiple A record results from DNS lookups) are called the vhost's *address set*.

Unless a `NameVirtualHost` directive is used for the exact IP address and port pair in the `VirtualHost` directive, Apache selects the best match only on the basis of the IP address (or wildcard) and port number. If there are multiple identical best matches, the first `VirtualHost` appearing in the configuration file will be selected.

If you want Apache to *further* discriminate on the basis of the HTTP Host header supplied by the client, the `NameVirtualHost` directive *must* appear with the exact IP address (or wildcard) and
port pair used in a corresponding set of VirtualHost directives.

The name-based virtual host selection occurs only after a single IP-based virtual host has been selected, and only considers the set of virtual hosts that carry an identical IP address and port pair.

Hostnames can be used in place of IP addresses in a virtual host definition, but it is resolved at startup and is not recommended.

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

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

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

```
NameVirtualHost 111.22.33.55
<VirtualHost 111.22.33.55>
  # server C
  ...
</VirtualHost>
NameVirtualHost 111.22.33.44
<VirtualHost 111.22.33.44>
  # server A
</VirtualHost>
```
(To aid the readability of your configuration you should prefer the left variant.)

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

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

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

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

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

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

If the main_server has no ServerName at this point, then the hostname of the machine that httpd is running on is used instead. We will call the main_server address set those IP addresses returned by a DNS lookup on the ServerName of the main_server.

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

Any vhost that includes the magic _default_ wildcard is given the same ServerName as the main_server.
The server determines which vhost to use for a request as follows:

**Hash table lookup**

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

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

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

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

**IP-based vhost**

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

**Name-based vhost**

If the entry corresponds to a name-based vhost the name list contains one or more vhost structures. This list contains the vhosts in the same order as the VirtualHost directives appear in the config file.
The first vhost on this list (the first vhost in the config file with the specified IP address) has the highest priority and catches any request to an unknown server name or a request without a Host: header field.

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

The complete list of names in the VirtualHost directive are treated just like a (non wildcard) ServerAlias (but are not overridden by any ServerAlias statement).

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

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

**Persistent connections**

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

**Absolute URI**
If the URI from the request is an absolute URI, and its hostname and port match the main server or one of the configured virtual hosts and match the address and port to which the client sent the request, then the scheme/hostname/port prefix is stripped off and the remaining relative URI is served by the corresponding main server or virtual host. If it does not match, then the URI remains untouched and the request is taken to be a proxy request.

Observations

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

- A _default_ vhost catches a request only if there is no other vhost with a matching IP address and a matching port number for the request. The request is only caught if the port number to which the client sent the request matches the port number of your _default_ vhost which is your standard Listen by default. A wildcard port can be specified (i.e., _default_:*) to catch requests to any available port. This also applies to NameVirtualHost * vhosts. Note that this is simply an extension of the "best match" principle, as a specific and exact match is favored over a wildcard.

- The main_server is only used to serve a request if the IP address and port number to which the client connected is unspecified and does not match any other vhost (including a _default_ vhost). In other words the main_server only catches a request for an unspecified address/port combination (unless there is a _default_ vhost which matches that port).

- A _default_ vhost or the main_server is never matched for a request with an unknown or missing Host: header field if the client connected to an address (and port) which is used for name-based vhosts, e.g., in a NameVirtualHost directive.

- You should never specify DNS names in VirtualHost directives because it will force your server to rely on DNS to boot. Furthermore it poses a security threat if you do not control the DNS for all the domains listed. There's more information available on this and the next two topics.

- ServerName should always be set for each vhost. Otherwise a DNS lookup is required for each vhost.
In addition to the tips on the [DNS Issues](https://httpd.apache.org/docs/2.4/dns.html) page, here are some further tips:

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

Apache

1. `setrlimit()`
2. `setrlimit(RLIMIT_NOFILE)` (Solaris 2.3)
3. 
4. `stdio 256`

: 

- `<VirtualHost>`
- `1 2` Apache

```
#!/bin/sh
ulimit -S -n 100
exec httpd
```
LogFormat  %v:

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

common log format ( ServerName ) ( Custom Log Formats )

(1) split-logfile Apache support

split-logfile < /logs/multiple_vhost_log.log
<table>
<thead>
<tr>
<th>DNS</th>
<th>Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>DNS A</td>
</tr>
</tbody>
</table>
<VirtualHost www.abc.dom>
ServerAdmin webgirl@abc.dom
DocumentRoot /www/abc
</VirtualHost>

Apache
Apache DNS

www.abc.dom 192.0.2.1

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

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

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

192.0.2.1   (http://www.abc.dom/whatever URL
    def.dom
Apache 1.1  Apache  httpd  IP
ServerName  C  gethostname(
DNS
DNS
OS
/getconf
/etc/nsswitch.conf
DNS
mod_env
HOSTRESORDER  local
CGI  man  OS
- VirtualHost IP
- Listen IP
- ServerName
- <VirtualHost _default_:*>
DNS Apache 1.2

DNS

DNS

DNS

DNS

DNS

IP DNS

HTTP/1.1 Host IP
SSL/TLS:

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

WebHTTPApache SSL
mod_ssl

The Open Group Research Institute Frederick J. Hirsch
Introducing SSL and Certificates using SSLeay Web
Engelschall (mod_ssl) [: Frederick Hirsch ()
Apache

"
(Certificate Authority CA)  
(certificate)  (authentication)  

1:

2:

<table>
<thead>
<tr>
<th>Field</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name ()</td>
<td>CN</td>
<td>CN=www.example.com</td>
</tr>
<tr>
<td>Organization or Company ()</td>
<td>O</td>
<td>O=Example Japan K.K.</td>
</tr>
<tr>
<td>Organizational Unit ()</td>
<td>OU</td>
<td>OU=Customer Service</td>
</tr>
<tr>
<td>City/Locality ()</td>
<td>L</td>
<td>L=Sapporo</td>
</tr>
<tr>
<td>State/Province ()</td>
<td>ST</td>
<td>ST=Hokkaido</td>
</tr>
<tr>
<td>Country()</td>
<td>C</td>
<td>ISO C=JP</td>
</tr>
</tbody>
</table>
ASN.1 Encoding Rules (DER) Basic Encoding Rules
Base64 [MIME] ASCII
"Privacy Enhanced Mail"

**PEM (example.crt)**

```plaintext
-----BEGIN CERTIFICATE-----
MIIC7jCCAAlegAwIBAgIBATANBgkqhkiG9w0BAQQFACCBqTELMAkGA1UEBhMCWFkx
FTATBgNVBAgTDFNuYWtlIER1c2VydETMTEGAEh4R8gYJ4UoAwIBAgIBATANBgkqhkiG9w0BAQ
-----END CERTIFICATE-----
```

(CA)
Thawte

VeriSign

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

SSL

## 4: SSL

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
<th>Vendor Standards</th>
<th>Compatible Browsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLv2.0</td>
<td>Vendor Standard (Netscape Corp.) [SSL2]</td>
<td>SSL</td>
<td>NS Navigator 1.x/2.x, MS IE 3.x, Lynx/2.8+OpenSSL</td>
</tr>
<tr>
<td>SSLv3.0</td>
<td>Expired Internet Draft (Netscape Corp.) [SSL3]</td>
<td>SSL3</td>
<td>NS Navigator 2.x/3.x/4.x, MS IE 3.x/4.x, Lynx/2.8+OpenSSL</td>
</tr>
<tr>
<td>TLSv1.0</td>
<td>Proposed Internet Standard (IETF) [TLS1]</td>
<td>MAC HMAC block padding 3.0</td>
<td>Lynx/2.8+OpenSSL</td>
</tr>
</tbody>
</table>

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

- Message Authentication Code (MAC)

Hellman
SSL:

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

HTTP

SSL  HTTP
HTTPS  URL
mod_ssl  Apache

http  https (443)
[AC96]  

[X208]  

[X509]  

[PKCS]  

[MIME]  

[SSL2]  

[SSL3]  
[TLS1]
SSL/TLS Strong Encryption: Compatibility

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

mod_ssl mostly provides a superset of the functionality of all the other solutions, so it's simple to migrate from one of the older modules to mod_ssl. The configuration directives and environment variable names used by the older SSL solutions vary from those used in mod_ssl; mapping tables are included here to give the equivalents used by mod_ssl.
The mapping between configuration directives used by Apache-SSL 1.x and mod_ssl 2.0.x is given in Table 1. The mapping from Sioux 1.x and Stronghold 2.x is only partial because of special functionality in these interfaces which mod_ssl doesn't provide.

**Table 1: Configuration Directive Mapping**

<table>
<thead>
<tr>
<th>Old Directive</th>
<th>mod_ssl Directive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Apache-SSL 1.x &amp; mod_ssl 2.0.x compatibility:</strong></td>
<td></td>
</tr>
<tr>
<td>SSLEnable</td>
<td>SSLEngine on</td>
</tr>
<tr>
<td>SSLDisable</td>
<td>SSLEngine off</td>
</tr>
<tr>
<td>SSLLogFile <code>file</code></td>
<td>-</td>
</tr>
<tr>
<td>SSLRequiredCiphers <code>spec</code></td>
<td>SSLCipherSuite <code>spec</code></td>
</tr>
<tr>
<td>SSLRequireCipher <code>c1</code> ...</td>
<td>SSLRequire <code>%{SSL_CIPHER} in {&quot;c1&quot;, ...}</code></td>
</tr>
<tr>
<td>SSLBanCipher <code>c1</code> ...</td>
<td>SSLRequire not (%{SSL_CIPHER} in {&quot;c1&quot;, ...})</td>
</tr>
<tr>
<td>SSLFakeBasicAuth</td>
<td>SSLOptions +FakeBasicAuth</td>
</tr>
<tr>
<td>SSLCacheServerPath <code>dir</code></td>
<td>-</td>
</tr>
<tr>
<td>SSLCacheServerPort <code>integer</code></td>
<td>-</td>
</tr>
<tr>
<td><strong>Apache-SSL 1.x compatibility:</strong></td>
<td></td>
</tr>
<tr>
<td>SSLExportClientCertificates</td>
<td>SSLOptions +ExportCert</td>
</tr>
<tr>
<td>SSLCacheServerRunDir <code>dir</code></td>
<td>-</td>
</tr>
<tr>
<td><strong>Sioux 1.x compatibility:</strong></td>
<td></td>
</tr>
<tr>
<td>SSL_CertFile <code>file</code></td>
<td>SSLCertificateFile <code>file</code></td>
</tr>
<tr>
<td>SSL_KeyFile <code>file</code></td>
<td>SSLCertificateKeyFile <code>file</code></td>
</tr>
<tr>
<td>SSL_CipherSuite <code>arg</code></td>
<td>SSLCipherSuite <code>arg</code></td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>SSL_X509VerifyDir arg</td>
<td>SSLCACertificatePath arg</td>
</tr>
<tr>
<td>SSL_Log file</td>
<td>-</td>
</tr>
<tr>
<td>SSL_Connect flag</td>
<td>SSLEngine flag</td>
</tr>
<tr>
<td>SSL_ClientAuth arg</td>
<td>SSLVerifyClient arg</td>
</tr>
<tr>
<td>SSL_X509VerifyDepth arg</td>
<td>SSLVerifyDepth arg</td>
</tr>
<tr>
<td>SSL_FetchKeyPhraseFrom arg</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SessionDir dir</td>
<td>-</td>
</tr>
<tr>
<td>SSLRequire expr</td>
<td>-</td>
</tr>
<tr>
<td>SSL_CertFileType arg</td>
<td>-</td>
</tr>
<tr>
<td>SSL_KeyFileType arg</td>
<td>-</td>
</tr>
<tr>
<td>SSL_X509VerifyPolicy arg</td>
<td>-</td>
</tr>
<tr>
<td>SSL_LogX509Attributes arg</td>
<td>-</td>
</tr>
</tbody>
</table>

**Stronghold 2.x compatibility:**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrongholdAccelerator engine</td>
<td>SSLCryptoDevice engine</td>
</tr>
<tr>
<td>StrongholdKey dir</td>
<td>-</td>
</tr>
<tr>
<td>StrongholdLicenseFile dir</td>
<td>-</td>
</tr>
<tr>
<td>SSLFlag flag</td>
<td>SSLEngine flag</td>
</tr>
<tr>
<td>SSLSessionLockFile file</td>
<td>SSLMutex file</td>
</tr>
<tr>
<td>SSLCipherList spec</td>
<td>SSLCipherSuite spec</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>RequireSSL</td>
<td>SSLRequireSSL</td>
</tr>
<tr>
<td>SSLErrorFile file</td>
<td>-</td>
</tr>
<tr>
<td>SSLRoot dir</td>
<td>-</td>
</tr>
<tr>
<td>SSL_CertificateLogDir dir</td>
<td>-</td>
</tr>
<tr>
<td>AuthCertDir dir</td>
<td>-</td>
</tr>
<tr>
<td>SSL_Group name</td>
<td>-</td>
</tr>
<tr>
<td>SSLProxyMachineCertPath dir</td>
<td>SSLProxyMachineCertPath dir</td>
</tr>
<tr>
<td>SSLProxyMachineCertFile file</td>
<td>SSLProxyMachineCertFile file</td>
</tr>
<tr>
<td>SSLProxyCipherList spec</td>
<td>SSLProxyCipherSpec spec</td>
</tr>
</tbody>
</table>
The mapping between environment variable names used by the older SSL solutions and the names used by mod_ssl is given in Table 2.

**Table 2: Environment Variable Derivation**

<table>
<thead>
<tr>
<th>Old Variable</th>
<th>mod_ssl Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_PROTOCOL_VERSION</td>
<td>SSL_PROTOCOL</td>
</tr>
<tr>
<td>SSLEAY_VERSION</td>
<td>SSL_VERSION_LIBRARY</td>
</tr>
<tr>
<td>HTTPS_SECRETKEYSIZE</td>
<td>SSL_CIPHER_USEKEYS</td>
</tr>
<tr>
<td>HTTPS_KEYSIZE</td>
<td>SSL_CIPHER_ALGKEYS</td>
</tr>
<tr>
<td>HTTPS_CIPHER</td>
<td>SSL_CIPHER</td>
</tr>
<tr>
<td>HTTPS_EXPORT</td>
<td>SSL_CIPHER_EXPORT</td>
</tr>
<tr>
<td>SSL_SERVER_KEY_SIZE</td>
<td>SSL_CIPHER_ALGKEYS</td>
</tr>
<tr>
<td>SSL_SERVER_CERTIFICATE</td>
<td>SSL_SERVER_CERT</td>
</tr>
<tr>
<td>SSL_SERVER_CERT_START</td>
<td>SSL_SERVER_V_START</td>
</tr>
<tr>
<td>SSL_SERVER_CERT_END</td>
<td>SSL_SERVER_V_END</td>
</tr>
<tr>
<td>SSL_SERVER_CERT_SERIAL</td>
<td>SSL_SERVER_M_SERIAL</td>
</tr>
<tr>
<td>SSL_SERVER_SIGNATURE_ALGORITHM</td>
<td>SSL_SERVER_A_SIG</td>
</tr>
<tr>
<td>SSL_SERVER_DN</td>
<td>SSL_SERVER_S_DN</td>
</tr>
<tr>
<td>SSL_SERVER_CN</td>
<td>SSL_SERVER_S_DN_CN</td>
</tr>
<tr>
<td>SSL_SERVER_EMAIL</td>
<td>SSL_SERVER_S_DN_Email</td>
</tr>
<tr>
<td>SSL_SERVER_O</td>
<td>SSL_SERVER_S_DN_O</td>
</tr>
<tr>
<td>SSL_SERVER_OU</td>
<td>SSL_SERVER_S_DN_OU</td>
</tr>
<tr>
<td>SSL_SERVER_C</td>
<td>SSL_SERVER_S_DN_C</td>
</tr>
<tr>
<td>SSL_SERVER_SP</td>
<td>SSL_SERVER_S_DN_SP</td>
</tr>
<tr>
<td>SSL_SERVER_L</td>
<td>SSL_SERVER_S_DN_L</td>
</tr>
<tr>
<td>SSL_SERVER_IDN</td>
<td>SSL_SERVER_I_DN</td>
</tr>
<tr>
<td>SSL_SERVER_ICN</td>
<td>SSL_SERVER_I_DN_CN</td>
</tr>
<tr>
<td>SSL_SERVER_IEMAIL</td>
<td>SSL_SERVER_I_DN_EMAIL</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>SSL_SERVER_IO</td>
<td>SSL_SERVER_I_DN_O</td>
</tr>
<tr>
<td>SSL_SERVER_IOU</td>
<td>SSL_SERVER_I_DN_OU</td>
</tr>
<tr>
<td>SSL_SERVER_IC</td>
<td>SSL_SERVER_I_DN_C</td>
</tr>
<tr>
<td>SSL_SERVER_ISP</td>
<td>SSL_SERVER_I_DN_SP</td>
</tr>
<tr>
<td>SSL_SERVER_IL</td>
<td>SSL_SERVER_I_DN_L</td>
</tr>
<tr>
<td>SSL_CLIENT_CERTIFICATE</td>
<td>SSL_CLIENT_CERT</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_START</td>
<td>SSL_CLIENT_V_START</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_END</td>
<td>SSL_CLIENT_V_END</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_SERIAL</td>
<td>SSL_CLIENT_M_SERIAL</td>
</tr>
<tr>
<td>SSL_CLIENT_SIGNATURE_ALGORITHM</td>
<td>SSL_CLIENT_A_SIG</td>
</tr>
<tr>
<td>SSL_CLIENT_CN</td>
<td>SSL_CLIENT_S_DN</td>
</tr>
<tr>
<td>SSL_CLIENT_CN</td>
<td>SSL_CLIENT_S_DN_CN</td>
</tr>
<tr>
<td>SSL_CLIENT_EMAIL</td>
<td>SSL_CLIENT_S_DN_Email</td>
</tr>
<tr>
<td>SSL_CLIENT_O</td>
<td>SSL_CLIENT_S_DN_O</td>
</tr>
<tr>
<td>SSL_CLIENT_OU</td>
<td>SSL_CLIENT_S_DN_OU</td>
</tr>
<tr>
<td>SSL_CLIENT_C</td>
<td>SSL_CLIENT_S_DN_C</td>
</tr>
<tr>
<td>SSL_CLIENT_SP</td>
<td>SSL_CLIENT_S_DN_SP</td>
</tr>
<tr>
<td>SSL_CLIENT_L</td>
<td>SSL_CLIENT_S_DN_L</td>
</tr>
<tr>
<td>SSL_CLIENT_IDN</td>
<td>SSL_CLIENT_I_DN</td>
</tr>
<tr>
<td>SSL_CLIENT_ICCN</td>
<td>SSL_CLIENT_I_DN_CN</td>
</tr>
<tr>
<td>SSL_CLIENT_IEMAIL</td>
<td>SSL_CLIENT_I_DN_Email</td>
</tr>
<tr>
<td>SSL_CLIENT_IO</td>
<td>SSL_CLIENT_I_DN_O</td>
</tr>
<tr>
<td>SSL_CLIENT_IOU</td>
<td>SSL_CLIENT_I_DN_OU</td>
</tr>
<tr>
<td>SSL_CLIENT_ICC</td>
<td>SSL_CLIENT_I_DN_C</td>
</tr>
<tr>
<td>SSL_CLIENT_ISP</td>
<td>SSL_CLIENT_I_DN_SP</td>
</tr>
<tr>
<td>SSL_CLIENT_ILI</td>
<td>SSL_CLIENT_I_DN_L</td>
</tr>
<tr>
<td>SSL_EXPORT</td>
<td>SSL_CIPHER_EXPORT</td>
</tr>
<tr>
<td>SSL_KEYSIZE</td>
<td>SSL_CIPHER_ALGKEYSIZE</td>
</tr>
<tr>
<td>SSL_SECKEYSIZE</td>
<td>SSL_CIPHER_USEKEYS</td>
</tr>
<tr>
<td>SSL_SSLEAY_VERSION</td>
<td>SSL_VERSION_LIBRARY</td>
</tr>
<tr>
<td>SSL_STRONG_CRYPTO</td>
<td>-</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---</td>
</tr>
<tr>
<td>SSL_SERVER_KEY_EXP</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_KEY_ALGORITHM</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_KEY_SIZE</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_SESSIONDIR</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_CERTIFICATELOGDIR</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_CERTFILE</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_KEYFILE</td>
<td>-</td>
</tr>
<tr>
<td>SSL_SERVER_KEYFILETYPE</td>
<td>-</td>
</tr>
<tr>
<td>SSL_CLIENT_KEY_EXP</td>
<td>-</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----</td>
</tr>
<tr>
<td>SSL_CLIENT_KEY_ALGORITHM</td>
<td>-</td>
</tr>
<tr>
<td>SSL_CLIENT_KEY_SIZE</td>
<td>-</td>
</tr>
</tbody>
</table>
When mod_ssl is enabled, additional functions exist for the Custom Log Format of mod_log_config as documented in the Reference Chapter. Beside the ``{%varname}x'' eXtension format function which can be used to expand any variables provided by any module, an additional Cryptography ``{%name}c'' cryptography format function exists for backward compatibility. The currently implemented function calls are listed in Table 3.

**Table 3: Custom Log Cryptography Function**

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%...{version}c</td>
<td>SSL protocol version</td>
</tr>
<tr>
<td>%...{cipher}c</td>
<td>SSL cipher</td>
</tr>
<tr>
<td>%...{subjectdn}c</td>
<td>Client Certificate Subject Distinguished Name</td>
</tr>
<tr>
<td>%...{issuerdn}c</td>
<td>Client Certificate Issuer Distinguished Name</td>
</tr>
<tr>
<td>%...{errcode}c</td>
<td>Certificate Verification Error (numerical)</td>
</tr>
<tr>
<td>%...{errstr}c</td>
<td>Certificate Verification Error (string)</td>
</tr>
</tbody>
</table>
SSL/TLS Strong Encryption: How-To

The solution to this problem is trivial and is left as an exercise for the reader.
-- Standard textbook cookie

How to solve particular security problems for an SSL-aware webserver is not always obvious because of the interactions between SSL, HTTP and Apache's way of processing requests. This chapter gives instructions on how to solve some typical situations. Treat it as a first step to find out the final solution, but always try to understand the stuff before you use it. Nothing is worse than using a security solution without knowing its restrictions and how it interacts with other systems.
How can I create a real SSLv2-only server?
The following creates an SSL server which speaks only the SSLv2 protocol and its ciphers:

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

How can I create an SSL server which accepts strong encryption only?
The following enables only the seven strongest ciphers:

```
httpd.conf
SSLProtocol all
SSLCipherSuite HIGH:MEDIUM
```

How can I create an SSL server which accepts strong encryption only, but allows export browsers to upgrade to stronger encryption?
This facility is called Server Gated Cryptography (SGC) and requires a Global ID server certificate, signed by a special CA.
certificate from Verisign. This enables strong encryption in 'export' versions of browsers, which traditionally could not support it (because of US export restrictions).

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

This can be done as follows:

```
httpd.conf
#
# allow all ciphers for the initial handshake,
# so export browsers can upgrade via SGC facility

<Directory /usr/local/apache2/htdocs>
# but finally deny all browsers which haven't upgraded
SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
</Directory>
```

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

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

<Location /strong/area>
# but https://hostname/strong/area/ and below
# requires strong ciphers
SSLCipherSuite HIGH:MEDIUM
</Location>
- How can I force clients to authenticate using certificates?
- How can I force clients to authenticate using certificates for a particular URL, but still allow arbitrary clients to access the rest of the server?
- How can I allow only clients who have certificates to access a particular URL, but allow all clients to access the rest of the server?
- How can I require HTTPS with strong ciphers, and either basic authentication or client certificates, for access to part of the Intranet website, for clients coming from the Internet?

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

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

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

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

To force clients to authenticate using certificates for a particular URL, you can use the per-directory reconfiguration features of `mod_ssl`:
How can I allow only clients who have certificates to access a particular URL, but allow all clients to access the rest of the server?

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

The `mod_auth_basic` method is generally required when the certificates are completely arbitrary, or when their DNs have no common fields (usually the organisation, etc.). In this case, you should establish a password database containing all clients allowed, as follows:
The password used in this example is the DES encrypted string "password". See the SSLOptions docs for more information.

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

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

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

<Directory /usr/local/apache2/htdocs/secure/area>
SSLVerifyClient require
SSLVerifyDepth 5
SSLOptions +FakeBasicAuth
SSLRequireSSL
SSLRequire %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"}
</Directory>
```

How can I require HTTPS with strong ciphers, and either basic authentication or client certificates, for access to part of the Intranet website, for clients coming from the Internet? I still want to allow plain HTTP access for clients on the Intranet.

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

httpd.conf

SSLCA CertificateFile conf/ssl.crt/company-ca.crt

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

<Directory /usr/local/apache2/htdocs/subarea>
  # Inside the subarea any Intranet access is allowed
  # but from the Internet only HTTPS + Strong-Cipher + Password
  # or the alternative HTTPS + Strong-Cipher + Client-Certificate
  
  # If HTTPS is used, make sure a strong cipher is used.
  # Additionally allow client certs as alternative to basic auth.
  SSLVerifyClient optional
  SSLVerifyDepth 1
  SSLOptions +FakeBasicAuth +StrictRequire
  SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
  
  # Force clients from the Internet to use HTTPS
  RewriteEngine on
  RewriteCond %{REMOTE_ADDR} !^192\.[0-9]+$ 
  RewriteCond %{HTTPS} !=on
  RewriteRule .* - [F]
  
  # Allow Network Access and/or Basic Auth
  Satisfy any
  
  # Network Access Control
  Order deny,allow
  Deny from all
  Allow 192.168.1.0/24
  
  # HTTP Basic Authentication
  AuthType basic
  AuthName "Protected Intranet Area"
  AuthBasicProvider file
  AuthUserFile conf/protected.passwd
  Require valid-user
</Directory>
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?

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

After US export restrictions on cryptographic software were loosened, mod_ssl became part of the Apache HTTP Server with the release of Apache httpd 2.

Is mod_ssl affected by the Wassenaar Arrangement?

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

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

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

So, both mod_ssl and OpenSSL are “in the public domain” for the purposes of the Wassenaar Arrangement and its “List of Dual Use Goods and Technologies And Munitions List”, and thus not affected by its provisions.
• Why do I get permission errors related to SSLMutex when I start Apache?

• Why does mod_ssl stop with the error "Failed to generate temporary 512 bit RSA private key" when I start Apache?

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

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

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

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

To prevent this error, mod_ssl has to provide enough entropy to the PRNG to allow it to work correctly. This can be done via the
SSLRandomSeed directive.
Is it possible to provide HTTP and HTTPS from the same server?

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

Which port does HTTPS use?

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

While you usually just use

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This rewrite ruleset lets you use hyperlinks of the form `<a href="document.html_SSL">`, to switch to HTTPS in a relative link. (Replace SSL with NOSSL to switch to HTTP.)
What are RSA Private Keys, CSRs and Certificates?

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

A Certificate Signing Request (CSR) is a digital file which contains your public key and your name. You send the CSR to a Certifying Authority (CA), who will convert it into a real Certificate, by signing it.
A Certificate contains your RSA public key, your name, the name of the CA, and is digitally signed by the CA. Browsers that know the CA can verify the signature on that Certificate, thereby obtaining your RSA public key. That enables them to send messages which only you can decrypt.

See the Introduction chapter for a general description of the SSL protocol.

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

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

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

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

1. Make sure OpenSSL is installed and in your PATH.

2. Run the following command, to create `server.key` and `server.crt` files:
   
   `$ openssl req -new -x509 -nodes -out server.crt -keyout server.key`
   
   These can be used as follows in your `httpd.conf` file:
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/
For details on how to create your own CA, and use this to sign a CSR, see below.

Once your CSR has been signed, you can see the details of the Certificate as follows:

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

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

```bash
SSLCertificateFile /path/to/this/server.crt
SSLCertificateKeyFile /path/to/this/server.key
```

The server.csr file is no longer needed.

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

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

1. Create a RSA private key for your server (will be Triple-DES encrypted and PEM formatted):
$ openssl genrsa -des3 -out server.key 1024

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

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

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

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

2. Create a self-signed Certificate (X509 structure) with the RSA key you just created (output will be PEM formatted):

$ openssl req -new -x509 -nodes -sha1 -days 365 -key server.key -out server.crt

This signs the server CSR and results in a server.crt file. You can see the details of this Certificate using:

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

How can I change the pass-phrase on my private key file?

You simply have to read it with the old pass-phrase and write it again, specifying the new pass-phrase. You can accomplish this with the following commands:
$ openssl rsa -des3 -in server.key -out server.key.new
$ mv server.key.new server.key

The first time you're asked for a PEM pass-phrase, you should enter the old pass-phrase. After that, you'll be asked again to enter a pass-phrase - this time, use the new pass-phrase. If you are asked to verify the pass-phrase, you'll need to enter the new pass-phrase a second time.

**How can I get rid of the pass-phrase dialog at Apache startup time?**

The reason this dialog pops up at startup and every re-start is that the RSA private key inside your server.key file is stored in encrypted format for security reasons. The pass-phrase is needed to decrypt this file, so it can be read and parsed. Removing the pass-phrase removes a layer of security from your server - proceed with caution!

1. Remove the encryption from the RSA private key (while keeping a backup copy of the original file):

   $ cp server.key server.key.org
   $ openssl rsa -in server.key.org -out server.key

2. Make sure the server.key file is only readable by root:

   $ chmod 400 server.key

Now server.key contains an unencrypted copy of the key. If you point your server at this file, it will not prompt you for a pass-
phrase. HOWEVER, if anyone gets this key they will be able to impersonate you on the net. PLEASE make sure that the permissions on this file are such that only root or the web server user can read it (preferably get your web server to start as root but run as another user, and have the key readable only by root).

As an alternative approach you can use the `SSLPassPhraseDialog exec:/path/to/program` facility.
Bear in mind that this is neither more nor less secure, of course.

**How do I verify that a private key matches its Certificate?**

A private key contains a series of numbers. Two of these numbers form the "public key", the others are part of the "private key". The "public key" bits are included when you generate a CSR, and subsequently form part of the associated Certificate.

To check that the public key in your Certificate matches the public portion of your private key, you simply need to compare these numbers. To view the Certificate and the key run the commands:

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

The `modulus' and the `public exponent' portions in the key and the Certificate must match. As the public exponent is usually 65537 and it's difficult to visually check that the long modulus numbers are the same, you can use the following approach:

```
$ openssl x509 -noout -modulus -in server.crt | openssl md5
$ openssl rsa -noout -modulus -in server.key | openssl md5
```

This leaves you with two rather shorter numbers to compare. It is,
in theory, possible that these numbers may be the same, without the modulus numbers being the same, but the chances of this are overwhelmingly remote.

Should you wish to check to which key or certificate a particular CSR belongs you can perform the same calculation on the CSR as follows:

```bash
$ openssl req -noout -modulus -in server.csr | openssl md5
```

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

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

Why does my 2048-bit private key not work?

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

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

The CA certificates under the path you configured with SSLCACertificatePath are found by SSLeay through hash symlinks. These hash values are generated by the `openssl
x509  -noout  -hash' command. However, the algorithm used to calculate the hash for a certificate changed between SSLeay 0.8 and 0.9. You will need to remove all old hash symlinks and create new ones after upgrading. Use the Makefile provided by `mod_ssl`.

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

The default certificate format for SSLeay/OpenSSL is PEM, which is simply Base64 encoded DER, with header and footer lines. For some applications (e.g. Microsoft Internet Explorer) you need the certificate in plain DER format. You can convert a PEM file `cert.pem` into the corresponding DER file `cert.der` using the following command: `$ openssl x509 -in cert.pem -out cert.der -outform DER`

**Why can't I find the getca or getverisign programs mentioned by Verisign, for installing my Verisign certificate?**

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

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

**Can I use the Server Gated Cryptography (SGC) facility (aka Verisign Global ID) with `mod_ssl`?**
Yes. mod_ssl has included support for the SGC facility since version 2.1. No special configuration is required - just use the Global ID as your server certificate. The step up of the clients is then automatically handled by mod_ssl at run-time.

**Why do browsers complain that they cannot verify my server certificate?**

One reason this might happen is because your server certificate is signed by an intermediate CA. Various CAs, such as Verisign or Thawte, have started signing certificates not with their root certificate but with intermediate certificates.

Intermediate CA certificates lie between the root CA certificate (which is installed in the browsers) and the server certificate (which you installed on the server). In order for the browser to be able to traverse and verify the trust chain from the server certificate to the root certificate it needs need to be given the intermediate certificates. The CAs should be able to provide you such intermediate certificate packages that can be installed on the server.

You need to include those intermediate certificates with the SSLCertificateChainFile directive.
Why do I get lots of random SSL protocol errors under heavy server load?

Why do I get lots of random SSL protocol errors under heavy server load?

Why do I get lots of random SSL protocol errors under heavy server load?

Why do I get lots of random SSL protocol errors under heavy server load?

Why do I get lots of random SSL protocol errors under heavy server load?
There can be a number of reasons for this, but the main one is problems with the SSL session Cache specified by the SSLSessionCache directive. The DBM session cache is the most likely source of the problem, so using the SHM session cache (or no cache at all) may help.

**Why does my webserver have a higher load, now that it serves SSL encrypted traffic?**

SSL uses strong cryptographic encryption, which necessitates a lot of number crunching. When you request a webpage via HTTPS, everything (even the images) is encrypted before it is transferred. So increased HTTPS traffic leads to load increases.

**Why do HTTPS connections to my server sometimes take up to 30 seconds to establish a connection?**

This is usually caused by a /dev/random device for SSLRandomSeed which blocks the read(2) call until enough entropy is available to service the request. More information is available in the reference manual for the SSLRandomSeed directive.

**What SSL Ciphers are supported by mod_ssl?**

Usually, any SSL ciphers supported by the version of OpenSSL in use, are also supported by mod_ssl. Which ciphers are available can depend on the way you built OpenSSL. Typically, at least the following ciphers are supported:

1. RC4 with MD5
2. RC4 with MD5 (export version restricted to 40-bit key)
3. RC2 with MD5
4. RC2 with MD5 (export version restricted to 40-bit key)
5. IDEA with MD5
6. DES with MD5
7. Triple-DES with MD5

To determine the actual list of ciphers available, you should run the following:

```
$ openssl ciphers -v
```

**Why do I get "no shared cipher" errors, when trying to use Anonymous Diffie-Hellman (ADH) ciphers?**

By default, OpenSSL does *not* allow ADH ciphers, for security reasons. Please be sure you are aware of the potential side-effects if you choose to enable these ciphers.

In order to use Anonymous Diffie-Hellman (ADH) ciphers, you must build OpenSSL with ``-DSSL_ALLOW_ADH``, and then add ``ADH`` into your SSLCipherSuite.

**Why do I get a 'no shared ciphers' error when connecting to my newly installed server?**

Either you have made a mistake with your SSLCipherSuite directive (compare it with the pre-configured example in httpd.conf-dist) or you chose to use DSA/DH algorithms instead of RSA when you generated your private key and ignored or overlooked the warnings. If you have chosen DSA/DH, then your server cannot communicate using RSA-based SSL ciphers (at least until you configure an additional RSA-based certificate/key pair). Modern browsers like NS or IE can only communicate over SSL using RSA ciphers. The result is the "no shared ciphers" error. To fix this, regenerate your server certificate/key pair, using the RSA algorithm.
Why can't I use SSL with name-based/non-IP-based virtual hosts?

The reason is very technical, and a somewhat "chicken and egg" problem. The SSL protocol layer stays below the HTTP protocol layer and encapsulates HTTP. When an SSL connection (HTTPS) is established Apache/mod_ssl has to negotiate the SSL protocol parameters with the client. For this, mod_ssl has to consult the configuration of the virtual server (for instance it has to look for the cipher suite, the server certificate, etc.). But in order to go to the correct virtual server Apache has to know the Host HTTP header field. To do this, the HTTP request header has to be read. This cannot be done before the SSL handshake is finished, but the information is needed in order to complete the SSL handshake phase. See the next question for how to circumvent this issue.

Note that if you have a wildcard SSL certificate, or a certificate that has multiple hostnames on it using subjectAltName fields, you can use SSL on name-based virtual hosts without further workarounds.

Why is it not possible to use Name-Based Virtual Hosting to identify different SSL virtual hosts?

Name-Based Virtual Hosting is a very popular method of identifying different virtual hosts. It allows you to use the same IP address and the same port number for many different sites. When people move on to SSL, it seems natural to assume that the same method can be used to have lots of different SSL virtual hosts on the same server.

It is possible, but only if using a 2.2.12 or later web server, built with 0.9.8j or later OpenSSL. This is because it requires a feature that only the most recent revisions of the SSL specification added, called Server Name Indication (SNI).

Note that if you have a wildcard SSL certificate, or a certificate that
has multiple hostnames on it using subjectAltName fields, you can use SSL on name-based virtual hosts without further workarounds.

The reason is that the SSL protocol is a separate layer which encapsulates the HTTP protocol. So the SSL session is a separate transaction, that takes place before the HTTP session has begun. The server receives an SSL request on IP address X and port Y (usually 443). Since the SSL request did not contain any Host: field, the server had no way to decide which SSL virtual host to use. Usually, it just used the first one it found which matched the port and IP address specified.

If you are using a version of the web server and OpenSSL that support SNI, though, and the client’s browser also supports SNI, then the hostname is included in the original SSL request, and the web server can select the correct SSL virtual host. You can, of course, use Name-Based Virtual Hosting to identify many non-SSL virtual hosts (all on port 80, for example) and then have a single SSL virtual host (on port 443). But if you do this, you must make sure to put the non-SSL port number on the NameVirtualHost directive, e.g.

```
NameVirtualHost 192.168.1.1:80
```

Other workaround solutions include:

- Using separate IP addresses for different SSL hosts. Using different port numbers for different SSL hosts.

**How do I get SSL compression working?**

Although SSL compression negotiation was defined in the specification of SSLv2 and TLS, it took until May 2004 for RFC 3749 to define DEFLATE as a negotiable standard compression method.
OpenSSL 0.9.8 started to support this by default when compiled with the zlib option. If both the client and the server support compression, it will be used. However, most clients still try to initially connect with an SSLv2 Hello. As SSLv2 did not include an array of preferred compression algorithms in its handshake, compression cannot be negotiated with these clients. If the client disables support for SSLv2, either an SSLv3 or TLS Hello may be sent, depending on which SSL library is used, and compression may be set up. You can verify whether clients make use of SSL compression by logging the %{SSL_COMPRESS_METHOD} 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:

```bash
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:

```bash
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.

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

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

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

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

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

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

Beginning with version 2.2.30, mod_ssl will use DH parameters which include primes with lengths of more than 1024 bits. Java 7 and earlier limit their support for DH prime sizes to a maximum of 1024 bits, however.

If your Java-based client aborts with exceptions such as java.lang.RuntimeException: Could not generate DH keypair and java.security.InvalidAlgorithmParameterException: Prime size must be multiple of 64, and can only range from 512 to 1024 (inclusive), and httpd logs tlsv1 alert internal error (SSL alert number 80) (at LogLevel info or higher), you can either rearrange mod_ssl's cipher list with SSLCipherSuite (possibly in conjunction with SSLHonorCipherOrder), or you can use custom DH parameters with a 1024-bit prime, which will always have precedence over any of the built-in DH parameters.

To generate custom DH parameters, use the openssl dhparam 1024 command. Alternatively, you can use the following standard 1024-bit DH parameters from RFC 2409, section 6.2:

```
-----BEGIN DH PARAMETERS-----
MIGHAoGBAP///////////yQ/aoiFowjTExmKLgNwc0SkCTgiKZ8x0Agu+pjsTmyJR
Sgh5jjQE3e+VGbPN0kMbMCsKbfJfFDr9tVtbnHCREsFtXZiXn7G9ExC6aY37WsL
/1y29Aa37e44a/taiZ+lrp8kEXxLH+ZJKGZR7OZTgf///////////AgEC
-----END DH PARAMETERS-----
```

Add the custom parameters including the "BEGIN DH PARAMETERS" and "END DH PARAMETERS" lines to the end of the first certificate file you have configured using the SSLCertificateFile directive.
What information resources are available in case of mod_ssl problems?

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). This information is required in order to find a reason for your core dump.

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

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

How do I get a backtrace, to help find the reason for my core dump?
Following are the steps you will need to complete, to get a backtrace:

1. Make sure you have debugging symbols available, at least in
Apache. On platforms where you use GCC/GDB, you will have to build Apache+mod_ssl with ```OPTIM=-g -ggdb3''` to get this. On other platforms at least ```OPTIM=-g''` is needed.

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

3. Analyze the core-dump. For this, run `gdb /path/to/httpd /tmp/httpd.core` or a similar command. In GDB, all you have to do then is to enter `bt`, and voila, you get the backtrace. For other debuggers consult your local debugger manual.
| mod_auth_basic | Allow |
| mod_authn_file | AuthGroupFile |
| mod_authz_groupfile | AuthName |
| mod_authz_host | AuthType |
| AuthUserFile | Deny |
| Options | Require |
<Directory .htaccess>

AllowOverride AuthConfig

</Directory>
/usr/local/apache/passwd

Apache htpasswd Apache

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

htpasswd

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

htpasswd
/usr/local/apache/bin/htpasswd

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

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

AuthType Basic
mod_auth_digest

AuthName Realm (:

Realm
Realm

AuthUserFile  htpasswd
mod_authn_dbm  AuthDBMUserFile
Apache

Require
GroupName: rbowen dpitts sungo rshersey

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

.htaccess

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

GroupName password

Require valid-user

require user rbowen
Basic
<table>
<thead>
<tr>
<th>Allow</th>
<th>Deny</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Allow from address</strong></td>
<td></td>
</tr>
<tr>
<td><em>address</em></td>
<td>IP</td>
</tr>
<tr>
<td><strong>Deny from 205.252.46.165</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Deny from host.example.com</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Deny from 192.101.205</strong></td>
<td><strong>Deny from cyberthugs.com moreidiots.com</strong></td>
</tr>
<tr>
<td><strong>Order</strong></td>
<td><strong>Deny</strong></td>
</tr>
<tr>
<td>Order deny,allow</td>
<td>Deny from all</td>
</tr>
<tr>
<td><strong>Allow</strong></td>
<td></td>
</tr>
</tbody>
</table>
mod au
Apache Tutorial: CGI
### CGI (Common Gateway Interface)

<table>
<thead>
<tr>
<th>mod_alias</th>
<th>AddHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_cgi</td>
<td>Options</td>
</tr>
<tr>
<td></td>
<td>ScriptAlias</td>
</tr>
</tbody>
</table>
```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/

<Directory /usr/local/apache2/htdocs/somedir>
  Options +ExecCGI
</Directory>
```
<Directory>
    Options +ExecCGI
    AddHandler cgi-script .cgi .pl
</Directory>

.htaccess
httpd.conf CGI

User

.cgi CGI

<Directory /home/*/public_html>
    Options +ExecCGI
    AddHandler cgi-script .cgi
</Directory>

cgi-bin CGI

<Directory /home/*/public_html/cgi-bin>
    Options ExecCGI
    SetHandler cgi-script
</Directory>
Content-type: text/html

Hello, World.

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

Hello, World. 1
chmod a+x first.pl

#!/usr/bin/perl
cd /usr/local/apache2/cgi-bin
./first.pl

(perl

Content-Type HTTP
end of script headers

Suexec

suexec script headers

suexec apachectl -V SUEXEC_BIN

suexec suexec

suexec suexec -V suexec
#!/usr/bin/perl

print "Content-type: text/html\n\n";

foreach $key (keys %ENV) {
    print "$key --> $ENV{$key}<br>
";
}

name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Squirrel%20Monkey
CGI
Perl  CGI  CPAN
C  CGI
CGI

HTML Writers Guide
http://www.hwg.org/lists/hwg-servers/

Apache

CGI

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0
Apache: Server Side Includes

HTML
<table>
<thead>
<tr>
<th>mod_include</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_cgi</td>
<td>XBitHack</td>
</tr>
<tr>
<td>mod_expires</td>
<td>AddType</td>
</tr>
<tr>
<td></td>
<td>SetOutputFilter</td>
</tr>
<tr>
<td></td>
<td>BrowserMatchNoCase</td>
</tr>
</tbody>
</table>

SSI  Server Side Includes
SSI
SSI  SSI
SSI (Server Side Includes) HTML
HTML
SSI
SSI  httpd.conf  .htaccess:

Options +Includes

SSI  Apache
Options

AddType text/html .shtml
AddOutputFilter INCLUDES .shtml

SSI

XBitHack:

XBitHack on

XBitHack chmod

chmod +x pagename.html

XBitHack

Windows

Apache SSI

1. XBitHack Full
2. mod_expires

SSI  Apache

.http  SSI

Apache SSI

HTTP

1. XBitHack Full
2. mod_expires

SSI
SSI:

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

HTML SSI

element

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

echo

CGI

config

<!--#config timefmt="%A %B %d, %Y" -->

Today is <!--#echo var="DATE_LOCAL" -->

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

timefmt

CGI

```
```

CGI SSI

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

? SSI

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

ssi.shtml

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

timefmt

strftime

/ file

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

LAST_MODIFIED
[an error occurred while processing this directive]

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

SSI

config

errmsg:

config

sizefmt

abbrev
CGI  SSI
Win32  DOS)

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

Windows

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

Windows

exec
IncludesNOEXEC  SSI  exec

°
Apache SSI

Apache 1.2

set

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

( LAST_MODIFIED

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

($)

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


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

endif

:

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


test_condition

: 

BrowserMatchNoCase macintosh Mac
BrowserMatchNoCase MSIE InternetExplorer

Macintosh

SSI:

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

Mac IE (JavaScript Mac)

( )
Apache : .htaccess

.htaccess
<table>
<thead>
<tr>
<th>mod_authn_file</th>
<th>core</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_authz_groupfile</td>
<td>mod_cgi</td>
</tr>
<tr>
<td>mod_include</td>
<td>mod_mime</td>
</tr>
<tr>
<td>AccessFileName</td>
<td>AllowOverride</td>
</tr>
<tr>
<td>Options</td>
<td>AddHandler</td>
</tr>
<tr>
<td>SetHandler</td>
<td>AuthType</td>
</tr>
<tr>
<td>AuthName</td>
<td>AuthUserFile</td>
</tr>
<tr>
<td>AuthGroupFile</td>
<td>Require</td>
</tr>
</tbody>
</table>
.htaccess

: .htaccess

AccessFileName .config

.htaccess

AllowOverride AddDefaultCharset .htaccess

FileInfo .htaccess

FileInfo

: 

:: ,,.htaccess
:: FileInfo

.htaccess ".htaccess"
.htaccess

ISP

.htaccess .htaccess

AllowOverride .htaccess Apache .htaccess

Apache

/www/htdocs/example Apache

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

.htaccess /www/htdocs/example

<Directory /www/htdocs/example>

/www/htdocs/example .htaccess:

/www/htdocs/example .htaccess

AddType text/example .exm
httpd.conf file

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

AllowOverride none .htaccess

AllowOverride None
.htaccess

Options +ExecCGI

(: .htaccess " Options"
 )

/www/htdocs/example1/example2 .htaccess:

Options Includes

/www/htdocs/example1/ example2 .htaccess:

CGI Options Includes

.htaccess

As discussed in the documentation on Configuration Sections, .htaccess files can override the <Directory> sections for the corresponding directory, but will be overridden by other types of configuration sections from the main configuration files. This fact can be used to enforce certain configurations, even in the presence of a liberal AllowOverride setting. For example, to prevent script execution while allowing anything else to be set in .htaccess you can use:

.htaccess <Directory>

AllowOverride
<Directory />
  Allowoverride All
</Directory>

<Location />
  Options +IncludesNoExec -ExecCGI
</Location>
AuthType Basic
AuthName "Password Required"
AuthUserFile /www/passwords/password.file
AuthGroupFile /www/passwords/group.file
Require Group admins

AllowOverride AuthConfig
.htaccess

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

AllowOverride Options
CGI

Options +ExecCGI
AddHandler cgi-script cgi pl

CGI

Options +ExecCGI
SetHandler cgi-script

AllowOverride Options
.htaccess

AllowOverride .htaccess

Apache
UserDir
"username" UserDir

URL
<table>
<thead>
<tr>
<th>mod_userdir</th>
<th>UserDir</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectoryMatch</td>
<td>AllowOverride</td>
</tr>
</tbody>
</table>
### UserDir

<table>
<thead>
<tr>
<th>UserDir</th>
<th>Path</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserDir enabled</td>
<td>UserDir disabled root jro fish</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>dissabled</strong></td>
<td><strong>UserDir</strong></td>
<td></td>
</tr>
<tr>
<td>UserDir disabled</td>
<td>UserDir enabled rbowen krietz</td>
<td></td>
</tr>
</tbody>
</table>

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

UserDir public_html

http://example.com/~rbowen/cgi-bin/example.cgi
Using Apache HTTP Server on Microsoft Windows

This document explains how to install, configure and run Apache 2.2 under Microsoft Windows. If you have questions after reviewing the documentation (and any event and error logs), you should consult the peer-supported users' mailing list.

This document assumes that you are installing a binary distribution of Apache. If you want to compile Apache yourself (possibly to help with development or tracking down bugs), see Compiling Apache for Microsoft Windows.
The primary Windows platform for running Apache 2.2 is Windows 2000 or later. The binary installer only works with the x86 family of processors, such as Intel and AMD processors. Always obtain and install the current service pack to avoid operating system bugs.

Running Apache on Windows 9x is ignored by the developers, and is strongly discouraged. On Windows NT 4.0, installing Service Pack 6 is required. Apache HTTP Server versions later than 2.2 will not run on any operating system earlier than Windows 2000.
The Apache HTTP Server Project itself does not provide binary releases of software, only source code. Individual committers may provide binary packages as a convenience, but it is not a release deliverable.

If you cannot compile the Apache HTTP Server yourself, you can obtain a binary package from numerous binary distributions available on the Internet.

Popular options for deploying Apache httpd, and, optionally, PHP and MySQL, on Microsoft Windows, include:

- [ApacheHaus](#)
- [Apache Lounge](#)
- [BitNami WAMP Stack](#)
- [WampServer](#)
- [XAMPP](#)
You need Microsoft Installer 2.0 or above for the installation to work. For Windows NT 4.0 and 2000 refer to Microsoft's article KB 292539. Windows XP and later do not require this update. The Windows 98/ME installer engine appears to no longer be available from Microsoft, and these instructions no longer detail such prerequisites.

Note that you cannot install two versions of Apache 2.2 on the same computer with the binary installer. You can, however, install a version of the 1.3 series and a version of the 2.2 series on the same computer without problems. If you need to have two different 2.2 versions on the same computer, you have to compile and install Apache from the source.

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

1. **Network Domain.** Enter the DNS domain in which your server is or will be registered in. For example, if your server's full DNS name is server.mydomain.net, you would type mydomain.net here.

2. **Server Name.** Your server's full DNS name. From the example above, you would type server.mydomain.net here.

3. **Administrator's Email Address.** Enter the server administrator's or webmaster's email address here. This address will be displayed along with error messages to the client by default.

4. **For whom to install Apache** Select for All Users, on Port 80, as a Service - Recommended if you'd like your new Apache to listen at port 80 for incoming traffic. It will
run as a service (that is, Apache will run even if no one is logged in on the server at the moment) Select only for the Current User, on Port 8080, when started Manually if you’d like to install Apache for your personal experimenting or if you already have another WWW server running on port 80.

5. **The installation type.** Select Typical for everything except the source code and libraries for module development. With Custom you can specify what to install. A full install will require about 13 megabytes of free disk space. This does not include the size of your web site(s).

6. **Where to install.** The default path is C:\Program Files\Apache Software Foundation under which a directory called Apache2.2 will be created by default.

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

Also, if you already have a file calledhtdocs\index.html, it will not be overwritten (and no index.html.default will be installed either). This means it should be safe to install Apache over an existing installation, although you would have to stop the existing running server before doing the installation, and then start the new one after the installation is finished.
After installing Apache, you must edit the configuration files in the conf subdirectory as required. These files will be configured during the installation so that Apache is ready to be run from the directory it was installed into, with the documents server from the subdirectory htdocs. There are lots of other options which you should set before you really start using Apache. However, to get started quickly, the files should work as installed.
One feature of the installer, "Build Headers and Libraries", can be ignored by most users, but should be installed if compiling third party modules. The "APR Iconv Code Pages" can similarly be omitted by most users, unless using mod_charset_lite or a third party module which relies on APR internationalization features.

There are requests to ship a .zip file from time to time. There is no point in the ASF consuming additional storage, mirroring and bandwidth for this purpose, because the .msi installer allows the installation artifacts to all be unpacked using the msiexec /a network installation option. Using this command against any .msi will result in an exploded tree of all of the individual files and components.

The installation options above can be customized by users familiar with msiexec.exe options and silent installation. The actual installer sources are available in the httpd/httpd/win32-msi/ tree of the httpd project subversion repository. For reference, some of the more common variables which may be modified are:

- **AgreeToLicense** (toggle to "Yes")
- **ALLUSERS** (choose between Null and "1")
- **ApplicationUsers** (toggle to "OnlyCurrentUser")
- **EXISTING_APACHE_SERVICE_PATH**
- **INSTALLDIR** (default "ProgramFilesFolder\Apache Software Foundation\Apache2.2")
- **INSTALLLEVEL** (default "100", refer to list below)
- **RESOLVED_WINSOCK2** (toggle to "1")
- **SERVERADMIN**
- **SERVERDOMAIN**
- **SERVERNAME**
- **SERVERPORT** (default "80")
• SERVERSSLPORT (default "443")
• SERVICEINTERNALNAME (default "Apache2.2", no spaces!)
• SERVICENAME (default "Apache2.2", include spaces)
• SetupType (default "Typical")

The installation level of various features, which may be individually toggled, include:

• Apache (1, base Apache HTTP Server 2.2 feature)
• ApacheDocs (11, Apache Documentation)
• ApacheMonitoring (41, Apache Service Taskbar Icon)
• ApacheRuntime (1, Apache Runtime)
• BuildFiles (101, Build Headers and Libraries)
• Iconv (21, APR Iconv Code Pages)
• OpensslBin (31, OpenSSL Runtime)
• SslBin (41, Ssl Binaries)
Apache is configured by the files in the `conf` subdirectory. These are the same files used to configure the Unix version, but there are a few different directives for Apache on Windows. See the directive index for all the available directives.

The main differences in Apache for Windows are:

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

The process management directives are also different:

**MaxRequestsPerChild**: Like the Unix directive, this controls how many requests (actually, connections) which a single child process will serve before exiting. However, unlike on Unix, a replacement process is not instantly available. Use the default `MaxRequestsPerChild 0`, unless instructed to change the behavior to overcome a memory leak in third party modules or in-process applications.

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

**ThreadsPerChild**: This directive is new. It tells the server how many threads it should use. This is the maximum number of connections the server can handle at once, so be sure to set this number high enough for your site if you get a lot of
hits. The recommended default is ThreadsPerChild 150, but this must be adjusted to reflect the greatest anticipated number of simultaneous connections to accept.

- The directives that accept filenames as arguments must use Windows filenames instead of Unix ones. However, because Apache may interpret backslashes as an "escape character" sequence, you should consistently use forward slashes in path names, not backslashes.

- While filenames are generally case-insensitive on Windows, URLs are still treated internally as case-sensitive before they are mapped to the filesystem. For example, the `<Location>`, `Alias`, and `ProxyPass` directives all use case-sensitive arguments. For this reason, it is particularly important to use the `<Directory>` directive when attempting to limit access to content in the filesystem, since this directive applies to any content in a directory, regardless of how it is accessed. If you wish to assure that only lowercase is used in URLs, you can use something like:

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

- When running, Apache needs write access only to the logs directory and any configured cache directory tree. Due to the issue of case insensitive and short 8.3 format names, Apache must validate all path names given. This means that each directory which Apache evaluates, from the drive root up to the directory leaf, must have read, list and traverse directory permissions. If Apache2.2 is installed at C:\Program Files, then the root directory, Program Files and Apache2.2 must all be visible to Apache.
- Apache for Windows contains the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the \Apache2.2\modules directory. To activate these or other modules, the new `LoadModule` directive must be used. For example, to activate the status module, use the following (in addition to the status-activating directives in access.conf):

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

  Information on [creating loadable modules](#) is also available.

- Apache can also load ISAPI (Internet Server Application Programming Interface) extensions such as those used by Microsoft IIS and other Windows servers. [More information is available](#). Note that Apache **cannot** load ISAPI Filters, and ISAPI Handlers with some Microsoft feature extensions will not work.

- When running CGI scripts, the method Apache uses to find the interpreter for the script is configurable using the `ScriptInterpreterSource` directive.

- Since it is often difficult to manage files with names like `.htaccess` in Windows, you may find it useful to change the name of this per-directory configuration file using the `AccessFilename` directive.

- Any errors during Apache startup are logged into the Windows event log when running on Windows NT. This mechanism acts as a backup for those situations where Apache is not yet prepared to use the `error.log` file. You can review the Windows Application Event Log by using the Event Viewer, e.g. Start - Settings - Control Panel - Administrative Tools - Event Viewer.
You can install Apache as a service automatically during the installation. If you chose to install for all users, the installation will create an Apache service for you. If you specify to install for yourself only, you can manually register Apache as a service after the installation. You have to be a member of the Administrators group for the service installation to succeed.

Apache comes with a utility called the Apache Service Monitor. With it you can see and manage the state of all installed Apache services on any machine on your network. To be able to manage an Apache service with the monitor, you have to first install the service (either automatically via the installation or manually).

You can install Apache as a Windows NT service as follows from the command prompt at the Apache bin subdirectory:

```
httpd.exe -k install
```

If you need to specify the name of the service you want to install, use the following command. You have to do this if you have several different service installations of Apache on your computer. If you specify a name during the install, you have to also specify it during any other -k operation.

```
httpd.exe -k install -n "MyServiceName"
```

If you need to have specifically named configuration files for different services, you must use this:

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

If you use the first command without any special parameters except -k install, the service will be called Apache2.2 and
the configuration will be assumed to be conf\httpd.conf.

Removing an Apache service is easy. Just use:

```plaintext
httpd.exe -k uninstall
```

The specific Apache service to be uninstalled can be specified by using:

```plaintext
httpd.exe -k uninstall -n "MyServiceName"
```

Normal starting, restarting and shutting down of an Apache service is usually done via the Apache Service Monitor, by using commands like NET START Apache2.2 and NET STOP Apache2.2 or via normal Windows service management. Before starting Apache as a service by any means, you should test the service's configuration file by using:

```plaintext
httpd.exe -n "MyServiceName" -t
```

You can control an Apache service by its command line switches, too. To start an installed Apache service you'll use this:

```plaintext
httpd.exe -k start -n "MyServiceName"
```

To stop an Apache service via the command line switches, use this:

```plaintext
httpd.exe -k stop -n "MyServiceName"
```

or

```plaintext
httpd.exe -k shutdown -n "MyServiceName"
```
You can also restart a running service and force it to reread its configuration file by using:

```
httpd.exe -k restart -n "MyServiceName"
```

By default, all Apache services are registered to run as the system user (the LocalSystem account). The LocalSystem account has no privileges to your network via any Windows-secured mechanism, including the file system, named pipes, DCOM, or secure RPC. It has, however, wide privileges locally.

**Never grant any network privileges to the LocalSystem account! If you need Apache to be able to access network resources, create a separate account for Apache as noted below.**

It is recommended that users create a separate account for running Apache service(s). If you have to access network resources via Apache, this is required.

1. Create a normal domain user account, and be sure to memorize its password.

2. Grant the newly-created user a privilege of Log on as a service and Act as part of the operating system. On Windows NT 4.0 these privileges are granted via User Manager for Domains, but on Windows 2000 and XP you probably want to use Group Policy for propagating these settings. You can also manually set these via the Local Security Policy MMC snap-in.

3. Confirm that the created account is a member of the Users group.

4. Grant the account read and execute (RX) rights to all document and script folders (htdocs and cgi-bin for
example).

5. Grant the account change (RWXD) rights to the Apache logs directory.

6. Grant the account read and execute (RX) rights to the httpd.exe binary executable.

It is usually a good practice to grant the user the Apache service runs as read and execute (RX) access to the whole Apache2.2 directory, except the logs subdirectory, where the user has to have at least change (RWXD) rights.

If you allow the account to log in as a user and as a service, then you can log on with that account and test that the account has the privileges to execute the scripts, read the web pages, and that you can start Apache in a console window. If this works, and you have followed the steps above, Apache should execute as a service with no problems.

**Error code 2186** is a good indication that you need to review the "Log On As" configuration for the service, since Apache cannot access a required network resource. Also, pay close attention to the privileges of the user Apache is configured to run as.

When starting Apache as a service you may encounter an error message from the Windows Service Control Manager. For example, if you try to start Apache by using the Services applet in the Windows Control Panel, you may get the following message:

```
Could not start the Apache2.2 service on \COMPUTER
Error 1067; The process terminated unexpectedly.
```

You will get this generic error if there is any problem with starting the Apache service. In order to see what is really causing the
problem you should follow the instructions for Running Apache for Windows from the Command Prompt.

If you are having problems with the service, it is suggested you follow the instructions below to try starting httpd.exe from a console window, and work out the errors before struggling to start it as a service again.
Running Apache as a service is usually the recommended way to use it, but it is sometimes easier to work from the command line (on Windows 9x running Apache from the command line is the recommended way due to the lack of reliable service support.)

To run Apache from the command line as a console application, use the following command:

```
httpd.exe
```

Apache will execute, and will remain running until it is stopped by pressing Control-C.

You can also run Apache via the shortcut Start Apache in Console placed to Start Menu --> Programs --> Apache HTTP Server 2.2.xx --> Control Apache Server during the installation. This will open a console window and start Apache inside it. If you don’t have Apache installed as a service, the window will remain visible until you stop Apache by pressing Control-C in the console window where Apache is running in. The server will exit in a few seconds. However, if you do have Apache installed as a service, the shortcut starts the service. If the Apache service is running already, the shortcut doesn’t do anything.

If Apache is running as a service, you can tell it to stop by opening another console window and entering:

```
httpd.exe -k shutdown
```

Running as a service should be preferred over running in a console window because this lets Apache end any current operations and clean up gracefully.

But if the server is running in a console window, you can only stop
it by pressing Control-C in the same window.

You can also tell Apache to restart. This forces it to reread the configuration file. Any operations in progress are allowed to complete without interruption. To restart Apache, either press Control-Break in the console window you used for starting Apache, or enter

```
httpd.exe -k restart
```

if the server is running as a service.

Note for people familiar with the Unix version of Apache: these commands provide a Windows equivalent to 

```
kil -TERM pid
```

and

```
kil -USR1 pid
```

The command line option used, -k, was chosen as a reminder of the kill command used on Unix.

If the Apache console window closes immediately or unexpectedly after startup, open the Command Prompt from the Start Menu --> Programs. Change to the folder to which you installed Apache, type the command httpd.exe, and read the error message. Then change to the logs folder, and review the error.log file for configuration mistakes. If you accepted the defaults when you installed Apache, the commands would be:

```
c: 
  cd "\Program Files\Apache Software Foundation\Apache2.2\bin"
  httpd.exe
```

Then wait for Apache to stop, or press Control-C. Then enter the following:

```
cd ..\logs
more < error.log
```
When working with Apache it is important to know how it will find the configuration file. You can specify a configuration file on the command line in two ways:

- `-f` specifies an absolute or relative path to a particular configuration file:

  ```
  httpd.exe -f "c:\my server files\anotherconfig.conf"
  ```

  or

  ```
  httpd.exe -f files\anotherconfig.conf
  ```

- `-n` specifies the installed Apache service whose configuration file is to be used:

  ```
  httpd.exe -n "MyServiceName"
  ```

In both of these cases, the proper `ServerRoot` should be set in the configuration file.

If you don't specify a configuration file with `-f` or `-n`, Apache will use the file name compiled into the server, such as `conf\httpd.conf`. This built-in path is relative to the installation directory. You can verify the compiled file name from a value labelled as `SERVER_CONFIG_FILE` when invoking Apache with the `-V` switch, like this:

```
httpd.exe -V
```

Apache will then try to determine its `ServerRoot` by trying the following, in this order:

1. A `ServerRoot` directive via the `-C` command line switch.
2. The -d switch on the command line.
3. Current working directory.
4. A registry entry which was created if you did a binary installation.
5. The server root compiled into the server. This is /apache by default, you can verify it by using httpd.exe -V and looking for a value labelled as HTTPD_ROOT.

During the installation, a version-specific registry key is created in the Windows registry. The location of this key depends on the type of the installation. If you chose to install Apache for all users, the key is located under the HKEY_LOCAL_MACHINE hive, like this (the version numbers will of course vary between different versions of Apache):

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\Apache\2.2.2
```

Correspondingly, if you chose to install Apache for the current user only, the key is located under the HKEY_CURRENT_USER hive, the contents of which are dependent of the user currently logged on:

```
HKEY_CURRENT_USER\SOFTWARE\Apache Software Foundation\Apache\2.2.2
```

This key is compiled into the server and can enable you to test new versions without affecting the current version. Of course, you must take care not to install the new version in the same directory as another version.

If you did not do a binary install, Apache will in some scenarios complain about the missing registry key. This warning can be ignored if the server was otherwise able to find its configuration file.
The value of this key is the `ServerRoot` directory which contains the `conf` subdirectory. When Apache starts it reads the `httpd.conf` file from that directory. If this file contains a `ServerRoot` directive which contains a different directory from the one obtained from the registry key above, Apache will forget the registry key and use the directory from the configuration file. If you copy the Apache directory or configuration files to a new location it is vital that you update the `ServerRoot` directive in the `httpd.conf` file to reflect the new location.
After starting Apache (either in a console window or as a service) it will be listening on port 80 (unless you changed the `Listen` directive in the configuration files or installed Apache only for the current user). To connect to the server and access the default page, launch a browser and enter this URL:

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

Apache should respond with a welcome page and you should see "It Works!". If nothing happens or you get an error, look in the `error.log` file in the `logs` subdirectory. If your host is not connected to the net, or if you have serious problems with your DNS (Domain Name Service) configuration, you may have to use this URL:

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

If you happen to be running Apache on an alternate port, you need to explicitly put that in the URL:

```plaintext
http://127.0.0.1:8080/
```

Once your basic installation is working, you should configure it properly by editing the files in the `conf` subdirectory. Again, if you change the configuration of the Windows NT service for Apache, first attempt to start it from the command line to make sure that the service starts with no errors.

Because Apache **cannot** share the same port with another TCP/IP application, you may need to stop, uninstall or reconfigure certain other services before running Apache. These conflicting services include other WWW servers, some firewall implementations, and even some client applications (such as Skype) which will use port
80 to attempt to bypass firewall issues.
Access to files over the network can be specified using two mechanisms provided by Windows:

**Mapped drive letters**
e.g., Alias /images/ Z:/

**UNC paths**
e.g., Alias /images/ //imagehost/www/images/

Mapped drive letters allow the administrator to maintain the mapping to a specific machine and path outside of the Apache httpd configuration. However, these mappings are associated only with interactive sessions and are not directly available to Apache httpd when it is started as a service. **Use only UNC paths for network resources in httpd.conf** so that the resources can be accessed consistently regardless of how Apache httpd is started. (Arcane and error prone procedures may work around the restriction on mapped drive letters, but this is not recommended.)

**Example DocumentRoot with UNC path**
```
DocumentRoot //dochost/www/html/
```

**Example DocumentRoot with IP address in UNC path**
```
DocumentRoot //192.168.1.50/docs/
```

**Example Alias and corresponding Directory with UNC path**
```
Alias /images/ //imagehost/www/images/

<Directory //imagehost/www/images/>
...
<Directory>
```

When running Apache httpd as a service, you must create a separate account in order to access network resources, as
described above.
If more than a few dozen piped loggers are used on an operating system instance, scaling up the "desktop heap" is often necessary. For more detailed information, refer to the piped logging documentation.
Compiling Apache for Microsoft Windows

There are many important points before you begin compiling Apache. See Using Apache with Microsoft Windows before you begin.
Compiling Apache requires the following environment to be properly installed:

- **Disk Space**

  Make sure you have at least 200 MB of free disk space available. After installation Apache requires approximately 80 MB of disk space, plus space for log and cache files, which can grow rapidly. The actual disk space requirements will vary considerably based on your chosen configuration and any third-party modules or libraries, especially when OpenSSL is also built. Because many files are text and very easily compressed, NTFS filesystem compression cuts these requirements in half.

- **Appropriate Patches**

  The httpd binary is built with the help of several patches to third party packages, which ensure the released code is buildable and debuggable. These patches are available and distributed from [http://www.apache.org/dist/httpd/binaries/win32/patches_applied/](http://www.apache.org/dist/httpd/binaries/win32/patches_applied/) and are recommended to be applied to obtain identical results as the "official" ASF distributed binaries.

- **Microsoft Visual C++ 6.0 (Visual Studio 97) or later.**

  Apache can be built using the command line tools, or from within the Visual Studio IDE Workbench. The command line build requires the environment to reflect the PATH, INCLUDE, LIB and other variables that can be configured with the `vcvars32.bat` script.

  You may want the Visual Studio Processor Pack for your
older version of Visual Studio, or a full (not Express) version of newer Visual Studio editions, for the ml.exe assembler. This will allow you to build OpenSSL, if desired, using the more efficient assembly code implementation.

Only the Microsoft compiler tool chain is actively supported by the active httpd contributors. Although the project regularly accepts patches to ensure MinGW and other alternative builds work and improve upon them, they are not actively maintained and are often broken in the course of normal development.

- Updated Microsoft Windows Platform SDK, February 2003 or later.

An appropriate Windows Platform SDK is included by default in the full (not express/lite) versions of Visual C++ 7.1 (Visual Studio 2002) and later, these users can ignore these steps unless explicitly choosing a newer or different version of the Platform SDK.

To use Visual C++ 6.0 or 7.0 (Studio 2000 .NET), the Platform SDK environment must be prepared using the setenv.bat script (installed by the Platform SDK) before starting the command line build or launching the msdev/devenv GUI environment. Installing the Platform SDK for Visual Studio Express versions (2003 and later) should adjust the default environment appropriately.

"c:\Program Files\Microsoft Visual Studio\VC98\Bin\VCVARS32"
"c:\Program Files\Platform SDK\setenv.bat"

- Perl and awk
Several steps recommended here require a perl interpreter during the build preparation process, but it is otherwise not required.

To install Apache within the build system, several files are modified using the awk.exe utility. awk was chosen since it is a very small download (compared with Perl or WSH/VB) and accomplishes the task of modifying configuration files upon installation. Brian Kernighan's [http://www.cs.princeton.edu/~bwk/btl.mirror/](http://www.cs.princeton.edu/~bwk/btl.mirror/) site has a compiled native Win32 binary, [http://www.cs.princeton.edu/~bwk/btl.mirror/awk95.exe](http://www.cs.princeton.edu/~bwk/btl.mirror/awk95.exe) which you must save with the name awk.exe (rather than awk95.exe).

If awk.exe is not found, Makefile.win's install target will not perform substitutions in the installed .conf files. You must manually modify the installed .conf files to allow the server to start. Search and replace all "@token@" tags as appropriate.

The Visual Studio IDE will only find awk.exe from the PATH, or executable path specified in the menu option Tools -> Options -> (Projects ->) Directories. Ensure awk.exe is in your system path.

Also note that if you are using Cygwin tools ([http://www.cygwin.com/](http://www.cygwin.com/)) the awk utility is named gawk.exe and that the file awk.exe is really a symlink to the gawk.exe file. The Windows command shell does not recognize symlinks, and because of this building InstallBin will fail. A workaround is to delete awk.exe from the cygwin installation and copy gawk.exe to awk.exe. Also note the
cygwin/mingw ports of gawk 3.0.x were buggy, please upgrade to 3.1.x before attempting to use any gawk port.

- [Optional] zlib library (for mod_deflate)

Zlib must be installed into a src/lib subdirectory named zlib. This must be built in-place. Zlib can be obtained from http://www.zlib.net/ -- the mod_deflate is confirmed to work correctly with version 1.2.3.

    nmake -f win32\Makefile.msc
    nmake -f win32\Makefile.msc test

- [Optional] OpenSSL libraries (for mod_ssl and ab.exe with ssl support)

The OpenSSL library is cryptographic software. The country in which you currently reside may have restrictions on the import, possession, use, and/or re-export to another country, of encryption software. BEFORE using any encryption software, please check your country's laws, regulations and policies concerning the import, possession, or use, and re-export of encryption software, to see if this is permitted. See http://www.wassenaar.org/ for more information.

Configuring and building OpenSSL requires perl to be installed.

OpenSSL must be installed into a src/lib subdirectory named openssl, obtained from http://www.openssl.org/source/, in order to compile mod_ssl or the abs.exe project, which is ab.c with SSL support enabled. To prepare OpenSSL to be linked to Apache
mod_ssl or abs.exe, and disable patent encumbered features in OpenSSL, you might use the following build commands:

```bash
perl Configure no-rc5 no-idea enable-mdc2 enable-zlib VC-WIN32 -Ipath/to/srclib/zlib -Lpath/to/srclib/zlib
ms\do_masm.bat
nmake -f ms\ntdll.mak
```

It is not advisable to use zlib-dynamic, as that transfers the cost of deflating SSL streams to the first request which must load the zlib dll. Note the suggested patch enables the -L flag to work with windows builds, corrects the name of zdll.lib and ensures .pdb files are generated for troubleshooting. If the assembler is not installed, you would add no-asm above and use ms\do_ms.bat instead of the ms\do_masm.bat script.

- [Optional] Database libraries (for mod_dbd and mod_authn_dbm)

The apr-util library exposes dbm (keyed database) and dbd (query oriented database) client functionality to the httpd server and its modules, such as authentication and authorization. The sdbm dbm and odbc dbd providers are compiled unconditionally.

The dbd support includes the Oracle instantclient package, MySQL, PostgreSQL and sqlite. To build these all, for example, set up the LIB to include the library path, INCLUDE to include the headers path, and PATH to include the dll bin path of all four SDK's, and set the DBD_LIST environment variable to inform the build which client driver SDKs are installed correctly, e.g.;

```bash
set DBD_LIST=sqlite3 pgsq1 oracle mysql
```
Similarly, the dbm support can be extended with DBM_LIST to
build a Berkeley DB provider (db) and/or gdbm provider, by
similarly configuring LIB, INCLUDE and PATH first to ensure
the client library libs and headers are available.
set DBM_LIST=db gdbm

Depending on the choice of database distributions, it may
be necessary to change the actual link target name (e.g.
gdbm.lib vs. libgdb.lib) that are listed in the corresponding
.dsp/.mak files within the directories srclib\apr-util\dbd or
...\dbm.
See the README-win32.txt file for more hints on obtaining
the various database driver SDKs.


Makefile.win is the top level Apache makefile. To compile Apache on Windows, simply use one of the following commands to build the release or debug flavor:

```
nmake /f Makefile.win _apacher
nmake /f Makefile.win _apached
```

Either command will compile Apache. The latter will disable optimization of the resulting files, making it easier to single step the code to find bugs and track down problems.

You can add your apr-util dbd and dbm provider choices with the additional make (environment) variables DBD_LIST and DBM_LIST, see the comments about [Optional] Database libraries, above. Review the initial comments in Makefile.win for additional options that can be provided when invoking the build.
Apache can also be compiled using VC++'s Visual Studio development environment. To simplify this process, a Visual Studio workspace, Apache .dsw, is provided. This workspace exposes the entire list of working .dsp projects that are required for the complete Apache binary release. It includes dependencies between the projects to assure that they are built in the appropriate order.

Open the Apache .dsw workspace, and select InstallBin (Release or Debug build, as desired) as the Active Project. InstallBin causes all related project to be built, and then invokes Makefile .win to move the compiled executables and dlls. You may personalize the INSTDIR= choice by changing InstallBin's Settings, General tab, Build command line entry. INSTDIR defaults to the /Apache2 directory. If you only want a test compile (without installing) you may build the BuildBin project instead.

The .dsp project files are distributed in Visual Studio 6.0 (98) format. Visual C++ 5.0 (97) will recognize them. Visual Studio 2002 (.NET) and later users must convert Apache .dsw plus the .dsp files into an Apache .sln plus .msproj files. Be sure you reconvert the .msproj file again if its source .dsp file changes! This is really trivial, just open Apache .dsw in the VC++ 7.0 IDE once again and reconvert.

There is a flaw in the .vcproj conversion of .dsp files. devenv.exe will mis-parse the /D flag for RC flags containing long quoted /D'efines which contain spaces. The command:

```
perl srclib\apr\build\cvtdsp.pl -2005
```
will convert the /D flags for RC flags to use an alternate, parseable syntax; unfortunately this syntax isn't supported by Visual Studio 97 or its exported .mak files. These /D flags are used to pass the long description of the mod_apachemodule.so files to the shared .rc resource version-identifier build.

Visual Studio 2002 (.NET) and later users should also use the Build menu, Configuration Manager dialog to uncheck both the Debug and Release Solution modules abs, mod_deflate and mod_ssl components, as well as every component starting with apr_db*. These modules are built by invoking nmake, or the IDE directly with the BinBuild target, which builds those modules conditionally if the srclib directories openssl and/or zlib exist, and based on the setting of DBD_LIST and DBM_LIST environment variables.
Exported .mak files pose a greater hassle, but they are required for Visual C++ 5.0 users to build mod_ssl, abs (ab with SSL support) and/or mod_deflate. The .mak files also support a broader range of C++ tool chain distributions, such as Visual Studio Express.

You must first build all projects in order to create all dynamic auto-generated targets, so that dependencies can be parsed correctly. Build the entire project from within the Visual Studio 6.0 (98) IDE, using the BuildAll target, then use the Project Menu Export for all makefiles (checking on "with dependencies"). Run the following command to correct absolute paths into relative paths so they will build anywhere:

```
perl srclib\apr\build\fixwin32mak.pl
```

You must type this command from the top level directory of the httpd source tree. Every .mak and .dep project file within the current directory and below will be corrected, and the timestamps adjusted to reflect the .dsp.

Always review the generated .mak and .dep files for Platform SDK or other local, machine specific file paths. The DevStudio\Common\MSDev98\bin\ (VC6) directory contains a sysincl.dat file, which lists all exceptions. Update this file (including both forward and backslashed paths, such as both sys/time.h and sys\time.h) to ignore such newer dependencies. Including local-install paths in a distributed .mak file will cause the build to fail completely.

If you contribute back a patch that revises project files, we must commit project files in Visual Studio 6.0 format. Changes should be simple, with minimal compilation and linkage flags that can be
recognized by all Visual Studio environments.
Once Apache has been compiled, it needs to be installed in its server root directory. The default is the \Apache2 directory, of the same drive.

To build and install all the files into the desired folder *dir* automatically, use one of the following nmake commands:

```
nmake /f Makefile.win installr INSTDIR=dir
nmake /f Makefile.win installd INSTDIR=dir
```

The *dir* argument to INSTDIR provides the installation directory; it can be omitted if Apache is to be installed into \Apache22 (of the current drive).
Note only the .dsp files are maintained between release builds. The .mak files are NOT regenerated, due to the tremendous waste of reviewer's time. Therefore, you cannot rely on the NMAKE commands above to build revised .dsp project files unless you then export all .mak files yourself from the project. This is unnecessary if you build from within the Microsoft Developer Studio environment.
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.
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.
There is no Apache install program for NetWare currently. If you are building Apache 2.0 for NetWare from source, you will need to copy the files over to the server manually.

Follow these steps to install Apache on NetWare from the binary download (assuming you will install to sys:/apache2):

- Unzip the binary download file to the root of the SYS: volume (may be installed to any volume)
- Edit the httpd.conf file setting ServerRoot and ServerName along with any file path values to reflect your correct server settings
- Add SYS:/APACHE2 to the search path, for example:

  SEARCH ADD SYS:\APACHE2

Follow these steps to install Apache on NetWare manually from your own build source (assuming you will install to sys:/apache2):

- Create a directory called Apache2 on a NetWare volume
- Copy APACHE2.NLM, APRLIB.NLM to SYS:/APACHE2
- Create a directory under SYS:/APACHE2 called BIN
- Copy HTDIGEST.NLM, HTPASSWD.NLM, HTDBM.NLM, LOGRES.NLM, R0TLOGS.NLM to SYS:/APACHE2/BIN
- Create a directory under SYS:/APACHE2 called CONF
- Copy the HTTPD-STD.CONF file to the SYS:/APACHE2/CONF directory and rename to HTTPD.CONF
- Copy the MIME.TYPES, CHARSET.CONV and MAGIC files to SYS:/APACHE2/CONF directory
- Copy all files and subdirectories in \HTTPD-2.0\DOCS\ICONS to SYS:/APACHE2/ICONS
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\MANUAL` to `SYS:/APACHE2/MANUAL`
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\ERROR` to `SYS:/APACHE2/ERROR`
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\DOCROOT` 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).
To start Apache just type `apache` at the console. This will load `apache` in the OS address space. If you prefer to load Apache in a protected address space you may specify the address space with the load statement as follows:

```
load address space = apache2 apache2
```

This will load Apache into an address space called `apache2`. Running multiple instances of Apache concurrently on NetWare is possible by loading each instance into its own protected address space.

After starting Apache, it will be listening to port 80 (unless you changed the `Listen` directive in the configuration files). To connect to the server and access the default page, launch a browser and enter the server's name or address. This should respond with a welcome page, and a link to the Apache manual. If nothing happens or you get an error, look in the `error_log` file in the `logs` directory.

Once your basic installation is working, you should configure it properly by editing the files in the `conf` directory.

To unload Apache running in the OS address space just type the following at the console:

```
unload apache2
```

or

```
apache2 shutdown
```

If `apache` is running in a protected address space specify the address space in the unload statement:
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.
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 module 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 requires MetroWerks CodeWarrior 6.x or higher. Once Apache has been built, it can be installed to the root of any NetWare volume. The default is the sys:/Apache2 directory.

Before running the server you must fill out the conf directory. Copy the file HTTPD-STD.CONF from the distribution conf directory and rename it to HTTPD.CONF. Edit the HTTPD.CONF file searching for all @@Value@@ markers and replacing them with the appropriate setting. Copy over the conf/magic and conf/mime.types files as well. Alternatively, a complete distribution can be built by including the keyword install when invoking the makefiles.

**Requirements:**

The following development tools are required to build Apache 2.0 for NetWare:

- Metrowerks CodeWarrior 6.0 or higher with the [NetWare PDK 3.0](http://developer.novell.com/ndk/apache.htm) or higher.
- [NetWare Libraries for C (LibC)](http://developer.novell.com/ndk/apache.htm)
- [LDAP Libraries for C](http://developer.novell.com/ndk/apache.htm)
- [ZLIB Compression Library source code](http://developer.novell.com/ndk/apache.htm)
- AWK utility (awk, gawk or similar). AWK can be downloaded from [http://developer.novell.com/ndk/apache.htm](http://developer.novell.com/ndk/apache.htm). The utility must be found in your windows path and must be named awk.exe.

**Building Apache using the NetWare makefiles:***
- Set the environment variable NOVELLLIBC to the location of the NetWare Libraries for C SDK, for example:

  ```
  Set NOVELLLIBC=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\cldapsdk\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
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 and \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 and chartables.c to the directory \httpd-2.0\os\netware on the build machine.
- Change directory to \httpd-2.0 and build Apache by
running "gmake -f nwgnumakefile". You can create a distribution directory by adding an install parameter to the command, for example:

```
gmake -f nwgnumakefile install
```

### Additional make options

- **gmake -f 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` and `\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` or `\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 recent OpenSSL 0.9.8 release source code from the [OpenSSL Source](http://www.openssl.org) page (older 0.9.7 versions need to be patched and are therefore not recommended).
- 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:
For performance reasons you should enable to build with ASM code. Download NASM from the SF site. Then configure OpenSSL to use ASM code:

```
Netware\build netware-libc nw-nasm enable-mdc2 enable-md5
```

Warning: dont use the CodeWarrior Assembler - it produces broken code!

- Before building Apache, set the environment variable OSSLSDK to the full path to the root of the openssl source code directory, and set WITH_MOD_SSL to 1.

```
Set OSSLSDK=d:\openssl-0.9.8x
Set WITH_MOD_SSL=1
```
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.

If folks are running Apache on a PA-8000 based system, they should consider "chatr'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 mpct1() 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.netperf.org/netperf/
The Apache EBCDIC Port

Warning: This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.
Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

(It is the SIEMENS family of mainframes running the BS2000/OSD operating system. This mainframe OS nowadays features a SVR4-derived POSIX subsystem).

The port was started initially to

- prove the feasibility of porting the Apache HTTP server to this platform
- find a "worthy and capable" successor for the venerable CERN-3.0 daemon (which was ported a couple of years ago), and to
- prove that Apache's preforking process model can on this platform easily outperform the accept-fork-serve model used by CERN by a factor of 5 or more.

This document serves as a rationale to describe some of the design decisions of the port to this machine.
One objective of the EBCDIC port was to maintain enough backwards compatibility with the (EBCDIC) CERN server to make the transition to the new server attractive and easy. This required the addition of a configurable method to define whether a HTML document was stored in ASCII (the only format accepted by the old server) or in EBCDIC (the native document format in the POSIX subsystem, and therefore the only realistic format in which the other POSIX tools like grep or sed could operate on the documents). The current solution to this is a "pseudo-MIME-format" which is intercepted and interpreted by the Apache server (see below). Future versions might solve the problem by defining an "ebcdic-handler" for all documents which must be converted.
Since all Apache input and output is based upon the BUFF data type and its methods, the easiest solution was to add the conversion to the BUFF handling routines. The conversion must be settable at any time, so a BUFF flag was added which defines whether a BUFF object has currently enabled conversion or not. This flag is modified at several points in the HTTP protocol:

- **set** before a request is received (because the request and the request header lines are always in ASCII format)
- **set/unset** when the request body is received - depending on the content type of the request body (because the request body may contain ASCII text or a binary file)
- **set** before a reply header is sent (because the response header lines are always in ASCII format)
- **set/unset** when the response body is sent - depending on the content type of the response body (because the response body may contain text or a binary file)
1. The relevant changes in the source are `#ifdef`'ed into two categories:

`#ifdef CHARSET_EBCDIC`

Code which is needed for any EBCDIC based machine. This includes character translations, differences in contiguity of the two character sets, flags which indicate which part of the HTTP protocol has to be converted and which part doesn't *etc*.

`#ifdef _OSD_POSIX`

Code which is needed for the SIEMENS BS2000/OSD mainframe platform only. This deals with include file differences and socket implementation topics which are only required on the BS2000/OSD platform.

2. The possibility to translate between ASCII and EBCDIC at the socket level (on BS2000 POSIX, there is a socket option which supports this) was intentionally *not* chosen, because the byte stream at the HTTP protocol level consists of a mixture of protocol related strings and non-protocol related raw file data. HTTP protocol strings are always encoded in ASCII (the GET request, any Header: lines, the chunking information *etc.*) whereas the file transfer parts (*i.e.*, GIF images, CGI output *etc.*) should usually be just "passed through" by the server. This separation between "protocol string" and "raw data" is reflected in the server code by functions like `bgets()` or `rvputs()` for strings, and functions like `bwrite()` for binary data. A global translation of everything would therefore be inadequate.

(In the case of text files of course, provisions must be made so that EBCDIC documents are always served in ASCII)
3. This port therefore features a built-in protocol level conversion for the server-internal strings (which the compiler translated to EBCDIC strings) and thus for all server-generated documents. The hard coded ASCII escapes \012 and \015 which are ubiquitous in the server code are an exception: they are already the binary encoding of the ASCII \n and \r and must not be converted to ASCII a second time. This exception is only relevant for server-generated strings; and external EBCDIC documents are not expected to contain ASCII newline characters.

4. By examining the call hierarchy for the BUFF management routines, I added an "ebcdic/ascii conversion layer" which would be crossed on every puts/write/get/gets, and a conversion flag which allowed enabling/disabling the conversions on-the-fly. Usually, a document crosses this layer twice from its origin source (a file or CGI output) to its destination (the requesting client): file -> Apache, and Apache -> client.

   The server can now read the header lines of a CGI-script output in EBCDIC format, and then find out that the remainder of the script's output is in ASCII (like in the case of the output of a WWW Counter program: the document body contains a GIF image). All header processing is done in the native EBCDIC format; the server then determines, based on the type of document being served, whether the document body (except for the chunking information, of course) is in ASCII already or must be converted from EBCDIC.

5. For Text documents (MIME types text/plain, text/html etc.), an implicit translation to ASCII can be used, or (if the users prefer to store some documents in raw ASCII form for faster serving, or because the files reside on a NFS-mounted directory tree) can be served without conversion.
Example:

to serve files with the suffix .ahtml as a raw ASCII text/html document without implicit conversion (and suffix .ascii as ASCII text/plain), use the directives:

```plaintext
AddType text/x-ascii-html .ahtml
AddType text/x-ascii-plain .ascii
```

Similarly, any text/foo MIME type can be served as "raw ASCII" by configuring a MIME type "text/x-ascii-foo" for it using AddType.

6. Non-text documents are always served "binary" without conversion. This seems to be the most sensible choice for, e.g., GIF/ZIP/AU file types. This of course requires the user to copy them to the mainframe host using the "rcp -b" binary switch.

7. Server parsed files are always assumed to be in native (i.e., EBCDIC) format as used on the machine, and are converted after processing.

8. For CGI output, the CGI script determines whether a conversion is needed or not: by setting the appropriate Content-Type, text files can be converted, or GIF output can be passed through unmodified. An example for the latter case is the wwwcount program which we ported as well.
Binary Files

All files with a Content-Type: which does not start with text/ are regarded as binary files by the server and are not subject to any conversion. Examples for binary files are GIF images, gzip-compressed files and the like.

When exchanging binary files between the mainframe host and a Unix machine or Windows PC, be sure to use the ftp "binary" (TYPE I) command, or use the rcp -b command from the mainframe host (the -b switch is not supported in unix rcp's).

Text Documents

The default assumption of the server is that Text Files (i.e., all files whose Content-Type: starts with text/) are stored in the native character set of the host, EBCDIC.

Server Side Included Documents

SSI documents must currently be stored in EBCDIC only. No provision is made to convert it from ASCII before processing.
<table>
<thead>
<tr>
<th>Module</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>core</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_access</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_actions</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_alias</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_asis</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_auth</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_auth_anon</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_auth_dbm</td>
<td>?</td>
<td>with own libdb.a</td>
</tr>
<tr>
<td>mod_autoindex</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_cern_meta</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>mod_cgi</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_digest</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_dir</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_so</td>
<td>-</td>
<td>no shared libs</td>
</tr>
<tr>
<td>mod_env</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_example</td>
<td>-</td>
<td>(test bed only)</td>
</tr>
<tr>
<td>mod_expires</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_headers</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_imagemap</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_include</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_info</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_log_agent</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_log_config</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_log_referer</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_mime</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_mime_magic</td>
<td>?</td>
<td>not ported yet</td>
</tr>
<tr>
<td>mod_negotiation</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>Status</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>mod_proxy</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_rewrite</td>
<td>+</td>
<td>untested</td>
</tr>
<tr>
<td>mod_setenvif</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_speling</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_status</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_unique_id</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_userdir</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>mod_usertrack</td>
<td>?</td>
<td>untested</td>
</tr>
<tr>
<td>Module</td>
<td>Status</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td><strong>JK</strong> (Formerly mod_jserv)</td>
<td>-</td>
<td>JAVA still being ported.</td>
</tr>
<tr>
<td>mod_php3</td>
<td>+</td>
<td>mod_php3 runs fine, with LDAP and GD and FreeType libraries.</td>
</tr>
<tr>
<td>mod_put</td>
<td>?</td>
<td>untested</td>
</tr>
<tr>
<td>mod_session</td>
<td>-</td>
<td>untested</td>
</tr>
</tbody>
</table>
httpd - Apache Hypertext Transfer Protocol Server

httpd is the Apache HyperText Transfer Protocol (HTTP) server program. It is designed to be run as a standalone daemon process. When used like this it will create a pool of child processes or threads to handle requests.

In general, httpd should not be invoked directly, but rather should be invoked via `apachectl` on Unix-based systems or as a service on Windows NT, 2000 and XP and as a console application on Windows 9x and ME.

See also

- Starting Apache httpd
- Stopping Apache httpd
- Configuration Files
- Platform-specific Documentation
- apachectl

▶
httpd [ -d serverroot ] [ -f config ] [ -C directive ] [ -c directive ] [ -D parameter ] [ -e level ] [ -E file ] [ -k start|restart|graceful|stop|graceful-stop ] [ -R directory ] [ -h ] [ -l ] [ -L ] [ -S ] [ -t ] [ -v ] [ -V ] [ -X ] [ -M ] [ -T ]

On Windows systems, the following additional arguments are available:

httpd [ -k install|config|uninstall ] [ -n name ] [ -w ]
-d serverroot
Set the initial value for the ServerRoot directive to serverroot. This can be overridden by the ServerRoot directive in the configuration file. The default is /usr/local/apache2.

-f config
Uses the directives in the file config on startup. If config does not begin with a /, then it is taken to be a path relative to the ServerRoot. The default is conf/httpd.conf.

-k start|restart|graceful|stop|graceful-stop
Signals httpd to start, restart, or stop. See Stopping Apache httpd for more information.

-C directive
Process the configuration directive before reading config files.

-c directive
Process the configuration directive after reading config files.

-D parameter
Sets a configuration parameter which can be used with <IfDefine> sections in the configuration files to conditionally skip or process commands at server startup and restart. Also can be used to set certain less-common startup parameters including -DNO_DETACH (prevent the parent from forking) and -DFOREGROUND (prevent the parent from calling setsid() et al).

-e level
Sets the LogLevel to level during server startup. This is useful for temporarily increasing the verbosity of the error messages to find problems during startup.

-E file
Send error messages during server startup to file.
-R `directory`
   When the server is compiled using the SHARED_CORE rule, this specifies the `directory` for the shared object files.

-h
   Output a short summary of available command line options.

-l
   Output a list of modules compiled into the server. This will **not** list dynamically loaded modules included using the `LoadModule` directive.

-L
   Output a list of directives together with expected arguments and places where the directive is valid.

-M
   Dump a list of loaded Static and Shared Modules.

-S
   Show the settings as parsed from the config file (currently only shows the virtualhost settings).

-T *(Available in 2.2.17 and later)*
   Skip document root check at startup/restart.

-t
   Run syntax tests for configuration files only. The program immediately exits after these syntax parsing tests with either a return code of 0 (Syntax OK) or return code not equal to 0 (Syntax Error). If `-D DUMP_VHOSTS` is also set, details of the virtual host configuration will be printed. If `-D DUMP_MODULES` is set, all loaded modules will be printed.

-v
   Print the version of `httpd`, and then exit.

-V
   Print the version and build parameters of `httpd`, and then
exit.

-X
Run httpd in debug mode. Only one worker will be started and the server will not detach from the console.

The following arguments are available only on the Windows platform:

-k install|config|uninstall
Install Apache httpd as a Windows NT service; change startup options for the Apache httpd service; and uninstall the Apache httpd service.

-n name
The name of the Apache httpd service to signal.

-w
Keep the console window open on error so that the error message can be read.
ab - Apache HTTP server benchmarking tool

ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.

See also

httpd
-A auth-username:password
Supply BASIC Authentication credentials to the server. The username and password are separated by a single \( : \) and sent on the wire base64 encoded. The string is sent regardless of whether the server needs it (\( i.e. \), has sent an 401 authentication needed).

-b windowsize
Size of TCP send/receive buffer, in bytes.

-c concurrency
Number of multiple requests to perform at a time. Default is one request at a time.

-C cookie-name=value
Add a Cookie: line to the request. The argument is typically in the form of a \( \text{name}=\text{value} \) pair. This field is repeatable.

-d
Do not display the "percentage served within XX [ms] table". (legacy support).

-e csv-file
Write a Comma separated value (CSV) file which contains for each percentage (from 1\% to 100\%) the time (in milliseconds) it took to serve that percentage of the requests. This is usually more useful than the 'gnuplot' file; as the results are already 'binned'.

-f protocol
Specify SSL/TLS protocol (SSL2, SSL3, TLS1, or ALL).

-g gnuplot-file
Write all measured values out as a 'gnuplot' or TSV (Tab separate values) file. This file can easily be imported into packages like Gnuplot, IDL, Mathematica, Igor or even Excel. The labels are on the first line of the file.
-h
  Display usage information.

-H custom-header
  Append extra headers to the request. The argument is typically in the form of a valid header line, containing a colon-separated field-value pair (i.e., "Accept-Encoding: zip/zop;8bit").

-i
  Do HEAD requests instead of GET.

-k
  Enable the HTTP KeepAlive feature, i.e., perform multiple requests within one HTTP session. Default is no KeepAlive.

-n requests
  Number of requests to perform for the benchmarking session. The default is to just perform a single request which usually leads to non-representative benchmarking results.

-p POST-file
  File containing data to POST. Remember to also set -T.

-P proxy-auth-username:password
  Supply BASIC Authentication credentials to a proxy en-route. The username and password are separated by a single : and sent on the wire base64 encoded. The string is sent regardless of whether the proxy needs it (i.e., has sent an 407 proxy authentication needed).

-q
  When processing more than 150 requests, ab outputs a progress count on stderr every 10% or 100 requests or so. The -q flag will suppress these messages.

-r
  Don't exit on socket receive errors.
-s
When compiled in (ab -h will show you) use the SSL protected https rather than the http protocol. This feature is experimental and very rudimentary. You probably do not want to use it.

-S
Do not display the median and standard deviation values, nor display the warning/error messages when the average and median are more than one or two times the standard deviation apart. And default to the min/avg/max values. (legacy support).

-t `timelimit`
Maximum number of seconds to spend for benchmarking. This implies a -n 50000 internally. Use this to benchmark the server within a fixed total amount of time. Per default there is no timelimit.

-T `content-type`
Content-type header to use for POST/PUT data, eg. application/x-www-form-urlencoded. Default: text/plain.

-u `PUT-file`
File containing data to PUT. Remember to also set -T.

-v `verbosity`
Set verbosity level - 4 and above prints information on headers, 3 and above prints response codes (404, 200, etc.), 2 and above prints warnings and info.

- V
Display version number and exit.

-w
Print out results in HTML tables. Default table is two columns wide, with a white background.
-x <table>-attributes
  String to use as attributes for <table>. Attributes are inserted <table here >.

-X proxy[:port]
  Use a proxy server for the requests.

-y <tr>-attributes
  String to use as attributes for <tr>.

-z <td>-attributes
  String to use as attributes for <td>.

-Z ciphersuite
  Specify SSL/TLS cipher suite (See openssl ciphers).
The following list describes the values returned by `ab`:

**Server Software**

The value, if any, returned in the `server` HTTP header of the first successful response. This includes all characters in the header from beginning to the point a character with decimal value of 32 (most notably: a space or CR/LF) is detected.

**Server Hostname**

The DNS or IP address given on the command line

**Server Port**

The port to which `ab` is connecting. If no port is given on the command line, this will default to 80 for http and 443 for https.

**SSL/TLS Protocol**

The protocol parameters negotiated between the client and server. This will only be printed if SSL is used.

**Document Path**

The request URI parsed from the command line string.

**Document Length**

This is size in bytes of the first successfully returning document. If the document length changes during testing, the response is considered an error.

**Concurrency Level**

The number of concurrent clients used during the test

**Time taken for tests**

This is the time taken from the moment the first socket connection is created to the moment the last response is received

**Complete requests**

The number of successful responses received

**Failed requests**
The number of requests that were considered a failure. If the number is greater than zero, another line will be printed showing the number of requests that failed due to connecting, reading, incorrect content length, or exceptions.

**Write errors**
The number of errors that failed during write (broken pipe).

**Non-2xx responses**
The number of responses that were not in the 200 series of response codes. If all responses were 200, this field is not printed.

**Keep-Alive requests**
The number of connections that resulted in Keep-Alive requests

**Total body sent**
If configured to send data as part of the test, this is the total number of bytes sent during the tests. This field is omitted if the test did not include a body to send.

**Total transferred**
The total number of bytes received from the server. This number is essentially the number of bytes sent over the wire.

**HTML transferred**
The total number of document bytes received from the server. This number excludes bytes received in HTTP headers

**Requests per second**
This is the number of requests per second. This value is the result of dividing the number of requests by the total time taken

**Time per request**
The average time spent per request. The first value is calculated with the formula concurrency * timetaken * 1000 / done while the second value is calculated with the
formula: \( \text{timetaken} \times 1000 \div \text{done} \)

**Transfer rate**
The rate of transfer as calculated by the formula \( \frac{\text{totalread}}{1024} \div \text{timetaken} \)
There are various statically declared buffers of fixed length. Combined with the lazy parsing of the command line arguments, the response headers from the server and other external inputs, this might bite you.

It does not implement HTTP/1.x fully; only accepts some 'expected' forms of responses. The rather heavy use of `strstr(3)` shows up top in profile, which might indicate a performance problem; i.e., you would measure the ab performance rather than the server's.
apachectl is a front end to the Apache HyperText Transfer Protocol (HTTP) server. It is designed to help the administrator control the functioning of the Apache httpd daemon.

The apachectl script can operate in two modes. First, it can act as a simple front-end to the httpd command that simply sets any necessary environment variables and then invokes httpd, passing through any command line arguments. Second, apachectl can act as a SysV init script, taking simple one-word arguments like start, restart, and stop, and translating them into appropriate signals to httpd.

If your Apache installation uses non-standard paths, you will need to edit the apachectl script to set the appropriate paths to the httpd binary. You can also specify any necessary httpd command line arguments. See the comments in the script for details.

The apachectl script returns a 0 exit value on success, and >0 if an error occurs. For more details, view the comments in the script.

See also

Starting Apache
Stopping Apache
Configuration Files
Platform Docs
httpd
When acting in pass-through mode, `apachectl` can take all the arguments available for the `httpd` binary.

`apachectl [ httpd-argument ]`

When acting in SysV init mode, `apachectl` takes simple, one-word commands, defined below.

`apachectl command`
Only the SysV init-style options are defined here. Other arguments are defined on the httpd manual page.

**start**
Start the Apache httpd daemon. Gives an error if it is already running. This is equivalent to apachectl -k start.

**stop**
Stops the Apache httpd daemon. This is equivalent to apachectl -k stop.

**restart**
Restarts the Apache httpd daemon. If the daemon is not running, it is started. This command automatically checks the configuration files as in configtest before initiating the restart to make sure the daemon doesn't die. This is equivalent to apachectl -k restart.

**fullstatus**
Displays a full status report from mod_status. For this to work, you need to have mod_status enabled on your server and a text-based browser such as lynx available on your system. The URL used to access the status report can be set by editing the STATUSURL variable in the script.

**status**
Displays a brief status report. Similar to the fullstatus option, except that the list of requests currently being served is omitted.

**graceful**
Gracefully restarts the Apache httpd daemon. If the daemon is not running, it is started. This differs from a normal restart in that currently open connections are not aborted. A side effect
is that old log files will not be closed immediately. This means that if used in a log rotation script, a substantial delay may be necessary to ensure that the old log files are closed before processing them. This command automatically checks the configuration files as in configtest before initiating the restart to make sure Apache doesn't die. This is equivalent to apachectl -k graceful.

**graceful-stop**

Gracefully stops the Apache httpd daemon. This differs from a normal stop in that currently open connections are not aborted. A side effect is that old log files will not be closed immediately. This is equivalent to apachectl -k graceful-stop.

**configtest**

Run a configuration file syntax test. It parses the configuration files and either reports Syntax Ok or detailed information about the particular syntax error. This is equivalent to apachectl -t.

The following option was available in earlier versions but has been removed.

**startssl**

To start httpd with SSL support, you should edit your configuration file to include the relevant directives and then use the normal apachectl start.
apxs - APache eXtenSion tool

apxs is a tool for building and installing extension modules for the Apache HyperText Transfer Protocol (HTTP) server. This is achieved by building a dynamic shared object (DSO) from one or more source or object files which then can be loaded into the Apache server under runtime via the `LoadModule` directive from `mod_so`.

So to use this extension mechanism your platform has to support the DSO feature and your Apache `httpd` binary has to be built with the `mod_so` module. The apxs tool automatically complains if this is not the case. You can check this yourself by manually running the command

```bash
$ httpd -l
```

The module `mod_so` should be part of the displayed list. If these requirements are fulfilled you can easily extend your Apache server's functionality by installing your own modules with the DSO mechanism by the help of this apxs tool:

```bash
$ apxs -i -a -c mod_foo.c
```

```bash
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]

```bash
$ apachectl restart
```

```bash
/path/to/apache/sbin/apachectl restart: httpd not running, trying to start
/path/to/apache/sbin/apachectl restart: httpd started
```

$ _

The arguments files can be any C source file (.c), a object file (.o) or even a library archive (.a). The apxs tool automatically recognizes
these extensions and automatically used the C source files for compilation while just using the object and archive files for the linking phase. But when using such pre-compiled objects make sure they are compiled for position independent code (PIC) to be able to use them for a dynamically loaded shared object. For instance with GCC you always just have to use -fpic. For other C compilers consult its manual page or at watch for the flags apxs uses to compile the object files.

For more details about DSO support in Apache read the documentation of mod_so or perhaps even read the src/modules/standard/mod_so.c source file.

See also

apachectl
httpd
apxs -g [ -S name=value ] -n modname

apxs -q [ -S name=value ] query ...

apxs -c [ -S name=value ] [ -o dsofile ] [ -I incdir ] [ -D name=value ] [ -L libdir ] [ -l libname ] [ -Wc,compiler-flags ] [ -Wl,linker-flags ] files ...

apxs -i [ -S name=value ] [ -n modname ] [ -a ] [ -A ] dso-file ...

apxs -e [ -S name=value ] [ -n modname ] [ -a ] [ -A ] dso-file ...
Common Options

-n modname
This explicitly sets the module name for the -i (install) and -g (template generation) option. Use this to explicitly specify the module name. For option -g this is required, for option -i the apxs tool tries to determine the name from the source or (as a fallback) at least by guessing it from the filename.

Query Options

-q
Performs a query for apxs's knowledge about certain settings. The query parameters can be one or more of the following strings: CC, CFLAGS, CFLAGS_SHLIB, INCLUDEDIR, LD_SHLIB, LDFLAGS_SHLIB, LIBEXECDIR, LIBS_SHLIB, SBINDIR, SYSCONFDIR, TARGET.
Use this for manually determining settings. For instance use

\[ \text{INC}=-I\`\text{apxs} -q \text{INCLUDEDIR}` \]

inside your own Makefiles if you need manual access to Apache's C header files.

Configuration Options

-S name=value
This option changes the apxs settings described above.

Template Generation Options

-g
This generates a subdirectory name (see option -n) and there two files: A sample module source file named mod_name.c
which can be used as a template for creating your own modules or as a quick start for playing with the apxs mechanism. And a corresponding Makefile for even easier build and installing of this module.

**DSO Compilation Options**

- **-c**
  This indicates the compilation operation. It first compiles the C source files (.c) of *files* into corresponding object files (.o) and then builds a dynamically shared object in *dsofile* by linking these object files plus the remaining object files (.o and .a) of *files*. If no -o option is specified the output file is guessed from the first filename in *files* and thus usually defaults to *mod_name.so*.

- **-o dsofile**
  Explicitly specifies the filename of the created dynamically shared object. If not specified and the name cannot be guessed from the *files* list, the fallback name *mod_unknown.so* is used.

- **-D name=value**
  This option is directly passed through to the compilation command(s). Use this to add your own defines to the build process.

- **-I incdir**
  This option is directly passed through to the compilation command(s). Use this to add your own include directories to search to the build process.

- **-L libdir**
  This option is directly passed through to the linker command. Use this to add your own library directories to search to the build process.
-l libname
   This option is directly passed through to the linker command. Use this to add your own libraries to search to the build process.

-Wc, compiler-flags
   This option passes compiler-flags as additional flags to the libtool --mode=compile command. Use this to add local compiler-specific options.

-Wl, linker-flags
   This option passes linker-flags as additional flags to the libtool --mode=link command. Use this to add local linker-specific options.

**DSO Installation and Configuration Options**

-i
   This indicates the installation operation and installs one or more dynamically shared objects into the server's modules directory.

-a
   This activates the module by automatically adding a corresponding LoadModule line to Apache's httpd.conf configuration file, or by enabling it if it already exists.

-A
   Same as option -a but the created LoadModule directive is prefixed with a hash sign (#), i.e., the module is just prepared for later activation but initially disabled.

-e
   This indicates the editing operation, which can be used with the -a and -A options similarly to the -i operation to edit Apache's httpd.conf configuration file without attempting to install the module.
Assume you have an Apache module named `mod_foo.c` available which should extend Apache’s server functionality. To accomplish this you first have to compile the C source into a shared object suitable for loading into the Apache server under runtime via the following command:

```
$ apxs -c mod_foo.c
/path/to/libtool --mode=compile gcc ... -c mod_foo.c
/path/to/libtool --mode=link gcc ... -o mod_foo.la mod_foo.slo
$ _
```

Then you have to update the Apache configuration by making sure a `LoadModule` directive is present to load this shared object. To simplify this step `apxs` provides an automatic way to install the shared object in its "modules" directory and updating the `httpd.conf` file accordingly. This can be achieved by running:

```
$ apxs -i -a mod_foo.la
/path/to/instdso.sh mod_foo.la /path/to/apache/modules
/path/to/libtool --mode=install cp mod_foo.la /path/to/apache/modules
/path/to/apache/modules ... chmod 755
/path/to/apache/modules/mod_foo.so
[activating module `foo' in /path/to/apache/conf/httpd.conf]
$ _
```

This way a line named

```plaintext
LoadModule foo_module modules/mod_foo.so
```

is added to the configuration file if still not present. If you want to have this disabled per default use the `-A` option, i.e.

```
$ apxs -i -A mod_foo.c
```

For a quick test of the `apxs` mechanism you can create a sample Apache module template plus a corresponding Makefile via:
Then you can immediately compile this sample module into a shared object and load it into the Apache server:

```bash
$ 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
$ _

Then you can immediately compile this sample module into a shared object and load it into the Apache server:

```bash
$ 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/instdso.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
/path/to/apache/sbin/apachectl restart: httpd started
$ _
```
configure - Configure the source tree

The configure script configures the source tree for compiling and installing the Apache HTTP Server on your particular platform. Various options allow the compilation of a server corresponding to your personal requirements.

This script, included in the root directory of the source distribution, is for compilation on Unix and Unix-like systems only. For other platforms, see the platform documentation.

See also

Compiling and Installing
You should call the configure script from within the root directory of the distribution.

`./configure [OPTION]... [VAR=VALUE]...`

To assign environment variables (e.g. CC, CFLAGS ...), specify them as `VAR=VALUE`. See below for descriptions of some of the useful variables.
Configuration options

The following options influence the behavior of configure itself.

-\texttt{C}

\texttt{--config-cache}

This is an alias for \texttt{--cache-file=config.cache}

\texttt{--cache-file=FILE}

The test results will be cached in file \textit{FILE}. This option is disabled by default.

-\texttt{h}

\texttt{--help \{short|recursive\}}

Output the help and exit. With the argument \texttt{short} only options specific to this package will displayed. The argument \texttt{recursive} displays the short help of all the included packages.

-\texttt{n}

\texttt{--no-create}

The configure script is run normally but does not create output files. This is useful to check the test results before generating makefiles for compilation.

-\texttt{q}

\texttt{--quiet}

Do not print checking \ldots{} messages during the configure
process.

--srcdir=DIR
Defines directory DIR to be the source file directory. Default is the directory where configure is located, or the parent directory.

--silent
Same as --quiet

-V
--version
Display copyright information and exit.

Installation directories
These options define the installation directory. The installation tree depends on the selected layout.

--prefix=PREFIX
Install architecture-independent files in PREFIX. By default the installation directory is set to /usr/local/apache2.

--exec-prefix=EPREFIX
Install architecture-dependent files in EPREFIX. By default the installation directory is set to the PREFIX directory.

By default, make install will install all the files in /usr/local/apache2/bin, /usr/local/apache2/lib etc. You can specify an installation prefix other than /usr/local/apache2 using --prefix, for instance --prefix=$HOME.

Define a directory layout
--enable-layout=LAYOUT
Configure the source code and build scripts to assume an
installation tree based on the layout LAYOUT. This allows you to separately specify the locations for each type of file within the Apache HTTP Server installation. The config.layout file contains several example configurations, and you can also create your own custom configuration following the examples. The different layouts in this file are grouped into <Layout FOO>... </Layout> sections and referred to by name as in FOO. The default layout is Apache.

Fine tuning of the installation directories
For better control of the installation directories, use the options below. Please note that the directory defaults are set by autoconf and are overwritten by the corresponding layout setting.

--bindir=DIR
Install user executables in DIR. The user executables are supporting programs like htpasswd, dbmmanage, etc. which are useful for site administrators. By default DIR is set to EPREFIX/bin.

--datadir=DIR
Install read-only architecture-independent data in DIR. By default datadir is set to PREFIX/share. This option is offered by autoconf and currently unused.

--includedir=DIR
Install C header files in DIR. By default includedir is set to EPREFIX/include.

--infodir=DIR
Install info documentation in DIR. By default infodir is set to PREFIX/info. This option is currently unused.

--libdir=DIR
Install object code libraries in DIR. By default libdir is set to
--libexecdir=DIR
Install the program executables (i.e., shared modules) in DIR. By default libexecdir is set to EPREFIX/modules.

--localstatedir=DIR
Install modifiable single-machine data in DIR. By default localstatedir is set to PREFIX/var. This option is offered by autoconf and currently unused.

--mandir=DIR
Install the man documentation in DIR. By default mandir is set to EPREFIX/man.

--oldincludedir=DIR
Install C header files for non-gcc in DIR. By default oldincludedir is set to /usr/include. This option is offered by autoconf and currently unused.

--sbindir=DIR
Install the system administrator executables in DIR. Those are server programs like httpd, apachectl, suexec, etc. which are neccessary to run the Apache HTTP Server. By default sbindir is set to EPREFIX/sbin.

--sharedstatedir=DIR
Install modifiable architecture-independent data in DIR. By default sharedstatedir is set to PREFIX/com. This option is offered by autoconf and currently unused.

--sysconfdir=DIR
Install read-only single-machine data like the server configuration files httpd.conf, mime.types, etc. in DIR. By default sysconfdir is set to PREFIX/conf.

System types
These options are used to cross-compile the Apache HTTP Server to run on another system. In normal cases, when building and running the server on the same system, these options are not used.

**--build=BUILD**
Defines the system type of the system on which the tools are being built. It defaults to the result of the script `config.guess`.

**--host=HOST**
Defines the system type of the system on which the server will run. `HOST` defaults to `BUILD`.

**--target=TARGET**
Configure for building compilers for the system type `TARGET`. It defaults to `HOST`. This option is offered by autoconf and not necessary for the Apache HTTP Server.

**Optional Features**
These options are used to fine tune the features your HTTP server will have.

**General syntax**
Generally you can use the following syntax to enable or disable a feature:

**--disable-**`FEATURE`**
Do not include `FEATURE`. This is the same as **--enable-**`FEATURE`=no.

**--enable-**`FEATURE`[`=``ARG`]**
Include `FEATURE`. The default value for `ARG` is yes.

**--enable-**`MODULE`=shared
The corresponding module will be build as DSO module.
--enable-MODULE=static
   By default enabled modules are linked statically. You can force this explicitly.

Note
configure will not complain about --enable-foo even if foo doesn't exist, so you need to type carefully.

Modules enabled by default
Some modules are compiled by default and have to be disabled explicitly. Use the following options to remove discrete modules from the compilation process.

--disable-actions
   Disable action triggering on requests, which is provided by mod_actions.

--disable-alias
   Disable the mapping of requests to different parts of the filesystem, which is provided by mod_alias.

--disable-asis
   Disable support for as-is filetypes, which is provided by mod_asis.

--disable-auth-basic
   Disable user-based access control provided by mod_auth_basic. This module provides for HTTP Basic Authentication, where the usernames and passwords are stored in plain text files.

--disable-autoindex
   Disable the directory listing functionality provided by mod_autoindex.

--disable-cgi
mod_cgi, which provides support for CGI scripts, is enabled by default when using a non-threaded MPM. Use this option to disable CGI support.

--disable-cgid
When using the threaded MPMs worker support for CGI scripts is provided by mod_cgid by default. To disable CGI support use this option.

--disable-charset-lite
Disable character set translation provided by mod_charset_lite. This module will be installed by default only on EBCDIC systems.

--disable-dir
Disable directory request handling provided by mod_dir.

--disable-env
Enable setting and clearing of environment variables, which is provided by mod_env.

--disable-http
Disable the HTTP protocol handling. The http module is a basic one, enabling the server to function as an HTTP server. It is only useful to disable it if you want to use another protocol module instead. **Don't disable this module unless you are really sure what you are doing.**
Note: This module will always be linked statically.

--disable-imagemap
Disable support for server based imagemaps, which provided by mod_imagemap.

--disable-include
Disable Server Side Includes provided by mod_include.

--disable-log-config
Disable the logging configuration provided by
mod_log_config. You won't be able to log requests to the server without this module.

--disable-mime
mod_mime associates the requested filename's extensions with the file's behavior and content (mime-type, language, character set and encoding). Disabling this module is normally not recommended.

--disable-negotiation
Disable content negotiation provided by mod_negotiation.

--disable-setenvif
Disable support for basing environment variables on headers, which is provided by mod_setenvif.

--disable-status
Enable the process/thread monitoring, which is provided by mod_status.

--disable-userdir
Disable the mapping of requests to user-specific directories, which is provided by mod_userdir.

Modules, disabled by default
Some modules are compiled by default and have to be enabled explicitly or by using the keywords most or all (see --enable-mods-shared below for further explanation) to be available. Therefore use the options below.

--enable-authn-anon
Enable anonymous user access provided by mod_authn_anon.

--enable-authn-dbms
mod_authn_dbm provides for HTTP Basic Authentication, where the usernames and passwords are stored in DBM type
database files. Use this option to enable the module.

### `--enable-authz-dbms`

mod_authz_dbm provides for HTTP Basic Authorization, where the usernames and groups are stored in DBM type database files. Use this option to enable the module.

### `--enable-auth-digest`

Enable RFC2617 Digest authentication provided by mod_auth_digest. This module uses plain text files to store the credentials.

### `--enable-authnz-ldap`

Enable LDAP based authentication provided by mod_authnz_ldap.

### `--enable-cache`

Enable dynamic file caching provided by mod_cache. This experimental module may be interesting for servers with high load or caching proxy servers. At least one storage management module (e.g. mod_disk_cache or mod_mem_cache) is also necessary.

### `--enable-cern-meta`

Enable the CERN-type meta files support provided by mod_cern_meta.

### `--enable-charset-lite`

Enable character set translation provided by mod_charset_lite. This module will be installed by default only on EBCDIC systems. On other systems, you have to enable it.

### `--enable-dav`

Enable the WebDAV protocol handling provided by mod_dav. Support for filesystem resources is provided by the separate module mod_dav_fs. This module is also automatically enabled with `--enable-dav`. 
Note: `mod_dav` can only be used together with the `http` protocol module.

`--enable-dav-fs`
Enable DAV support for filesystem resources, which is provided by `mod_dav_fs`. This module is a provider for the `mod_dav` module, so you should also use `--enable-dav`.

`--enable-dav-lock`
Enable `mod_dav_lock` which provides generic DAV locking support for backend modules. This module needs at least `mod_dav` to function, so you should also use `--enable-dav`.

`--enable-deflate`
Enable deflate transfer encoding provided by `mod_deflate`.

`--enable-disk-cache`
Enable disk caching provided by `mod_disk_cache`.

`--enable-expires`
Enable Expires header control provided by `mod_expires`.

`--enable-ext-filter`
Enable the external filter support provided by `mod_ext_filter`.

`--enable-file-cache`
Enable the file cache provided by `mod_file_cache`.

`--enable-headers`
Enable control of HTTP headers provided by `mod_headers`.

`--enable-info`
Enable the server information provided by `mod_info`.

`--enable-ldap`
Enable LDAP caching and connection pooling services provided by `mod_ldap`.
--enable-logio
Enable logging of input and output bytes including headers provided by mod_logio.

--enable-mem-cache
Enable memory caching provided by mod_mem_cache.

--enable-mime-magic
Enable automatical determining of MIME types, which is provided by mod_mime_magic.

--enable-isapi
Enable the isapi extension support provided by mod_isapi.

--enable-proxy
Enable the proxy/gateway functionality provided by mod_proxy. The proxying capabilities for AJP13, CONNECT, FTP, HTTP and the balancer are provided by the separate modules mod_proxy_ajp, mod_proxy_connect, mod_proxy_ftp, mod_proxy_http and mod_proxy_balancer. These five modules are also automatically enabled with --enable-proxy.

--enable-proxy-ajp
Enable proxy support for AJP13 (Apache JServ Protocol 1.3) request handling, which is provided by mod_proxy_ajp. This module is an extension for the mod_proxy module, so you should also use --enable-proxy.

--enable-proxy-balancer
Enable load balancing support for the AJP13, FTP and HTTP protocols, which is provided by mod_proxy_balancer. This module is an extension for the mod_proxy module, so you should also use --enable-proxy.

--enable-proxy-connect
Enable proxy support for CONNECT request handling, which is provided by `mod_proxy_connect`. This module is an extension for the `mod_proxy` module, so you should also use `--enable-proxy`.

`--enable-proxy-ftp`
Enable proxy support for FTP requests, which is provided by `mod_proxy_ftp`. This module is an extension for the `mod_proxy` module, so you should also use `--enable-proxy`.

`--enable-proxy-http`
Enable proxy support for HTTP requests, which is provided by `mod_proxy_http`. This module is an extension for the `mod_proxy` module, so you should also use `--enable-proxy`.

`--enable-rewrite`
Enable rule based URL manipulation provided by `mod_rewrite`.

`--enable-so`
Enable DSO capability provided by `mod_so`. This module will be automatically enabled if you use the `--enable-mods-shared` option.

`--enable-speling`
Enable the functionality to correct common URL misspellings, which is provided by `mod_speling`.

`--enable-ssl`
Enable support for SSL/TLS provided by `mod_ssl`.

`--enable-unique-id`
Enable the generation of per-request unique ids, which is provided by `mod_unique_id`.

`--enable-usertrack`
Enable user-session tracking provided by `mod_usertrack`.

`--enable-vhost-alias`
Enable mass virtual hosting provided by `mod_vhost_alias`.

**Modules for developers**
The following modules are useful only for developers and testing purposes and are disabled by default. Use the following options to enable them. If you are not sure whether you need one of these modules, omit them.

`--enable-bucketeer`
Enable the manipulation filter for buckets, which is provided by `mod_bucketeer`.

`--enable-case-filter`
Enable the example uppercase conversion output filter support of `mod_case_filter`.

`--enable-case-filter-in`
Enable the example uppercase conversion input filter support of `mod_case_filter_in`.

`--enable-echo`
Enable the ECHO server provided by `mod_echo`.

`--enable-example`
Enable the example and demo module `mod_example`.

`--enable-optional-fn-export`
Enable the example for an optional function exporter, which is provided by `mod_optional_fn_export`.

`--enable-optional-fn-import`
Enable the example for an optional function importer, which is provided by `mod_optional_fn_import`.

`--enable-optional-hook-export`
Enable the example for an optional hook exporter, which is
--enable-optional-hook-import
Enable the example optional hook importer, which is provided by mod_optional_hook_import.

MPMs and third-party modules
To add the necessary Multi Processing Module and additional third-party modules use the following options:

--with-module=module-type:module-file[, module-type:module-file]
Add one or more third-party modules to the list of statically linked modules. The module source file module-file will be searched in the modules/module-type subdirectory of your Apache HTTP server source tree. If it is not found there configure is considering module-file to be an absolute file path and tries to copy the source file into the module-type subdirectory. If the subdirectory doesn't exist it will be created and populated with a standard Makefile.in.

This option is useful to add small external modules consisting of one source file. For more complex modules you should read the vendor's documentation.

Note
If you want to build a DSO module instead of a statically linked use apxs.

--with-mpm=MPM
Choose the process model for your server. You have to select exactly one Multi-Processing Module. Otherwise the default MPM for your operating system will be taken. Possible MPMs are beos, mpmt_os2, prefork, and worker.
Cumulative and other options

--enable-maintainer-mode
Turn on debugging and compile time warnings.

--enable-mods-shared=MODULE-LIST
Defines a list of modules to be enabled and build as dynamic shared modules. This mean, these module have to be loaded dynamically by using the `LoadModule` directive.

`MODULE-LIST` is a space separated list of modulenames enclosed by quotation marks. The module names are given without the preceding mod_. For example:

```
--enable-mods-shared='headers rewrite dav'
```

Additionally you can use the special keywords all and most. For example,

```
--enable-mods-shared=most
```

will compile most modules and build them as DSO modules.

**Caveat:** `--enable-mods-shared=all` does not actually build all modules. To build all modules then, one might use:

```
./configure \
  --with-ldap \
  --enable-mods-shared="all ssl ldap cache proxy authn_alias mem_cache file_cache authnz_ldap charset_lite dav_lock disk_cache"
```

--enable-modules=MODULE-LIST
This option behaves similar to `--enable-mods-shared`, but will link the given modules statically. This mean, these modules will always be present while running `httpd`. They need not be loaded with `LoadModule`. 
--enable-v4-mapped
Allow IPv6 sockets to handle IPv4 connections.

--with-port=PORT
This defines the port on which httpd will listen. This port number is used when generating the configuration file httpd.conf. The default is 80.

--with-program-name
Define an alternative executable name. The default is httpd.

Optional packages
These options are used to define optional packages.

General syntax
Generally you can use the following syntax to define an optional package:

--with-PACKAGE[=ARG]
Use the package PACKAGE. The default value for ARG is yes.

--without-PACKAGE
Do not use the package PACKAGE. This is the same as --with-PACKAGE=no. This option is provided by autoconf but not very useful for the Apache HTTP Server.

Specific packages
--with-apr=DIR|FILE
The Apache Portable Runtime (APR) is part of the httpd source distribution and will automatically be build together with the HTTP server. If you want to use an already installed APR instead you have to tell configure the path to the apr-config script. You may set the absolute path and name or the directory to the installed APR. apr-config must exist
within this directory or the subdirectory bin.

--with-apr-util=DIR|FILE
The Apache Portable Runtime Utilities (APU) are part of the httpd source distribution and will automatically be build together with the HTTP server. If you want to use an already installed APU instead you have to tell configure the path to the apu-config script. You may set the absolute path and name or the directory to the installed APU. apu-config must exist within this directory or the subdirectory bin.

--with-ssl=DIR
If mod_ssl has been enabled configure searches for an installed OpenSSL. You can set the directory path to the SSL/TLS toolkit instead.

--with-z=DIR
configure searches automatically for an installed zlib library if your source configuration requires one (e.g., when mod_deflate is enabled). You can set the directory path to the compression library instead.

Several features of the Apache HTTP Server, including mod_authn_dbm, mod_authz_dbm, and mod_rewrite's DBM RewriteMap use simple key/value databases for quick lookups of information. SDBM is included in the APU, so this database is always available. If you would like to use other database types, use the following options to enable them:

--with-gdbm[=path]
If no path is specified, configure will search for the include files and libraries of a GNU DBM installation in the usual search paths. An explicit path will cause configure to look in path/lib and path/include for the relevant files. Finally, the path may specify specific include and library paths
separated by a colon.

--with-ndbm[=path]
Like --with-gdbm, but searches for a New DBM installation.

--with-berkeley-db[=path]
Like --with-gdbm, but searches for a Berkeley DB installation.

Note
The DBM options are provided by the APU and passed through to its configuration script. They are useless when using an already installed APU defined by --with-apr-util.

You may use more than one DBM implementation together with your HTTP server. The appropriately configured DBM type will be configured within the runtime configuration at each time.

Options for support programs

--enable-static-support
Build a statically linked version of the support binaries. This means, a stand-alone executable will be built with all the necessary libraries integrated. Otherwise the support binaries are linked dynamically by default.

--enable-suexec
Use this option to enable suexec, which allows you to set uid and gid for spawned processes. **Do not use this option unless you understand all the security implications of running a suid binary on your server.** Further options to configure suexec are described below.

It is possible to create a statically linked binary of a single support program by using the following options:
--enable-static-ab
   Build a statically linked version of ab.

--enable-static-checkgid
   Build a statically linked version of checkgid.

--enable-static-htdbm
   Build a statically linked version of htdbm.

--enable-static-htdigest
   Build a statically linked version of htdigest.

--enable-static-htpasswd
   Build a statically linked version of htpasswd.

--enable-static-logresolve
   Build a statically linked version of logresolve.

--enable-static-rotatelogs
   Build a statically linked version of rotatelogs.

suexec configuration options
The following options are used to fine tune the behavior of suexec. See Configuring and installing suEXEC for further information.

--with-suexec-bin
   This defines the path to suexec binary. Default is --sbindir (see Fine tuning of installation directories).

--with-suexec-caller
   This defines the user allowed to call suexec. It should be the same as the user under which httpd normally runs.

--with-suexec-docroot
   This defines the directory tree under which suexec access is allowed for executables. Default value is --datadir/htdocs.
--with-suexec-gidmin
Define this as the lowest GID allowed to be a target user for suexec. The default value is 100.

--with-suexec-logfile
This defines the filename of the suexec logfile. By default the logfile is named suexec_log and located in --logfiledir.

--with-suexec-safepath
Define the value of the environment variable PATH to be set for processes started by suexec. Default value is /usr/local/bin:/usr/bin:/bin.

--with-suexec-userdir
This defines the subdirectory under the user's directory that contains all executables for which suexec access is allowed. This setting is necessary when you want to use suexec together with user-specific directories (as provided by mod_userdir). The default is public_html.

--with-suexec-uidmin
Define this as the lowest UID allowed to be a target user for suexec. The default value is 100.

--with-suexec-umask
Set umask for processes started by suexec. It defaults to your system settings.
There are some useful environment variables to override the choices made by `configure` or to help it to find libraries and programs with nonstandard names or locations.

**CC**
Define the C compiler command to be used for compilation.

**CFLAGS**
Set C compiler flags you want to use for compilation.

**CPP**
Define the C preprocessor command to be used.

**CPPFLAGS**
Set C/C++ preprocessor flags, e.g. `-I#include_dir` if you have headers in a nonstandard directory `include_dir`.

**LDFLAGS**
Set linker flags, e.g. `-L/libdir` if you have libraries in a nonstandard directory `libdir`.

---

Copyright 2017 The Apache Software Foundation.
Licensed under the Apache License, Version 2.0.

Modules | Directives | FAQ | Glossary | Sitemap
**dbmmanage - Manage user authentication files in DBM format**

dbmmanage is used to create and update the DBM format files used to store usernames and password for basic authentication of HTTP users via `mod_authn_dbm`. Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by dbmmanage. This program can only be used when the usernames are stored in a DBM file. To use a flat-file database see `htpasswd`.

This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in `httpd` see the httpd manual, which is part of the Apache distribution or can be found at [http://httpd.apache.org/](http://httpd.apache.org/).

**See also**

- [httpd](http://httpd.apache.org/)
- `mod_authn_dbm`
- `mod_authz_dbm`
**Synopsis**

**dbmmanage**  
[ `encoding` ]  
filename  
add|adduser|check|delete|update username [  
encpasswd [ group[,]group...] [ comment ] ] ]

**dbmmanage** filename view [ username ]

**dbmmanage** filename import
filename
The filename of the DBM format file. Usually without the extension .db, .pag, or .dir.

username
The user for which the operations are performed. The username may not contain a colon (:).

encpasswd
This is the already encrypted password to use for the update and add commands. You may use a hyphen (-) if you want to get prompted for the password, but fill in the fields afterwards. Additionally when using the update command, a period (.) keeps the original password untouched.

group
A group, which the user is member of. A groupname may not contain a colon (:). You may use a hyphen (-) if you don't want to assign the user to a group, but fill in the comment field. Additionally when using the update command, a period (.) keeps the original groups untouched.

comment
This is the place for your opaque comments about the user, like realname, mailaddress or such things. The server will ignore this field.

Encodings
-d
crypt encryption (default, except on Win32, Netware)
-m
MD5 encryption (default on Win32, Netware)
-s
SHA1 encryption

-p
plaintext (not recommended)

**Commands**

**add**

Adds an entry for *username* to *filename* using the encrypted password *encpasswd*.

```
dbmmanage passwords.dat add rbowen foKntnEF3KSXA
```

**adduser**

Asks for a password and then adds an entry for *username* to *filename*.

```
dbmmanage passwords.dat adduser krietz
```

**check**

Asks for a password and then checks if *username* is in *filename* and if it's password matches the specified one.

```
dbmmanage passwords.dat check rbowen
```

**delete**

Deletes the *username* entry from *filename*.

```
dbmmanage passwords.dat delete rbowen
```

**import**

Reads *username:password* entries (one per line) from STDIN and adds them to *filename*. The passwords already have to be crypted.

**update**
Same as the adduser command, except that it makes sure `username` already exists in `filename`.

```
dbmmanage passwords.dat update rbowen
```

**view**

Just displays the contents of the DBM file. If you specify a `username`, it displays the particular record only.

```
dbmmanage passwords.dat view
```
One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, the GNU project's GDBM, and Berkeley DB 2. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by `filename` is the same format that dbmmanage expects to see. dbmmanage currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

`dbmmanage` has a list of DBM format preferences, defined by the `@AnyDBM::ISA` array near the beginning of the program. Since we prefer the Berkeley DB 2 file format, the order in which dbmmanage will look for system libraries is Berkeley DB 2, then NDBM, then GDBM and then SDBM. The first library found will be the library dbmmanage will attempt to use for all DBM file transactions. This ordering is slightly different than the standard `@AnyDBM::ISA` ordering in Perl, as well as the ordering used by the simple `dbmopen()` call in Perl, so if you use any other utilities to manage your DBM files, they must also follow this preference ordering. Similar care must be taken if using programs in other languages, like C, to access these files.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.
htcacheclean is used to keep the size of mod_disk_cache's storage within a certain limit. This tool can run either manually or in daemon mode. When running in daemon mode, it sleeps in the background and checks the cache directories at regular intervals for cached content to be removed. You can stop the daemon cleanly by sending it a TERM or INT signal.

See also

mod_disk_cache
htcacheclean [ -D ] [ -v ] [ -t ] [ -r ] [ -n ] -p path -l limit

htcacheclean [ -n ] [ -t ] [ -i ] -d interval -p path -l limit
-d

delement

Daemonize and repeat cache cleaning every interval minutes. This option is mutually exclusive with the -D, -v and -r options. To shutdown the daemon cleanly, just send it a SIGTERM or SIGINT.

-D

Do a dry run and don't delete anything. This option is mutually exclusive with the -d option.

-v

Be verbose and print statistics. This option is mutually exclusive with the -d option.

-r

Clean thoroughly. This assumes that the Apache web server is not running (otherwise you may get garbage in the cache). This option is mutually exclusive with the -d option and implies the -t option.

-n

Be nice. This causes slower processing in favour of other processes. htcacheclean will sleep from time to time so that (a) the disk IO will be delayed and (b) the kernel can schedule other processes in the meantime.

-t

Delete all empty directories. By default only cache files are removed, however with some configurations the large number of directories created may require attention. If your configuration requires a very large number of directories, to the point that inode or file allocation table exhaustion may become an issue, use of this option is advised.

-p

path

Specify path as the root directory of the disk cache. This
should be the same value as specified with the CacheRoot directive.

-**limit**
  Specify limit as the total disk cache size limit. The value is expressed in bytes by default (or attaching B to the number). Attach K for Kbytes or M for MBytes.

-**i**
  Be intelligent and run only when there was a modification of the disk cache. This option is only possible together with the -d option.
htcacheclean returns a zero status ("true") if all operations were successful, 1 otherwise.
htdbm - Manipulate DBM password databases

htdbm is used to manipulate the DBM format files used to store usernames and password for basic authentication of HTTP users via mod_authn_dbm. See the dbmmanage documentation for more information about these DBM files.

See also

httpd
dbmmanage
mod_authn_dbm
htdbm [ -TDBTYPE ] [ -c ] [ -m | -d | -p | -s ] [ -t ] [ -v ] [ -x ] filename username

htdbm -b [ -TDBTYPE ] [ -c ] [ -m | -d | -p | -s ] [ -t ] [ -v ] filename username password

htdbm -n [ -c ] [ -m | -d | -p | -s ] [ -t ] [ -v ] username

htdbm -nb [ -c ] [ -m | -d | -p | -s ] [ -t ] [ -v ] username password

htdbm -v [ -TDBTYPE ] [ -c ] [ -m | -d | -p | -s ] [ -t ] [ -v ] filename username

htdbm -vb [ -TDBTYPE ] [ -c ] [ -m | -d | -p | -s ] [ -t ] [ -v ] filename username password

htdbm -x [ -TDBTYPE ] [ -m | -d | -p | -s ] filename username

htdbm -l [ -TDBTYPE ]
-b
Use batch mode; i.e., get the password from the command line rather than prompting for it. This option should be used with extreme care, since the password is clearly visible on the command line.

-c
Create the passwdfile. If passwdfile already exists, it is rewritten and truncated. This option cannot be combined with the -n option.

-n
Display the results on standard output rather than updating a database. This option changes the syntax of the command line, since the passwdfile argument (usually the first one) is omitted. It cannot be combined with the -c option.

-m
Use MD5 encryption for passwords. On Windows, Netware and TPF, this is the default.

-d
Use crypt( ) encryption for passwords. The default on all platforms but Windows, Netware and TPF. Though possibly supported by htdbm on all platforms, it is not supported by the httpd server on Windows, Netware and TPF.

-s
Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (ldif).

-p
Use plaintext passwords. Though htdbm will support creation on all platforms, the httpd daemon will only accept plain text passwords on Windows, Netware and TPF.
-l
Print each of the usernames and comments from the database on stdout.

-t
Interpret the final parameter as a comment. When this option is specified, an additional string can be appended to the command line; this string will be stored in the "Comment" field of the database, associated with the specified username.

-v
Verify the username and password. The program will print a message indicating whether the supplied password is valid. If the password is invalid, the program exits with error code 3.

-x
Delete user. If the username exists in the specified DBM file, it will be deleted.

filename
The filename of the DBM format file. Usually without the extension .db, .pag, or .dir. If -c is given, the DBM file is created if it does not already exist, or updated if it does exist.

username
The username to create or update in passwdfile. If username does not exist in this file, an entry is added. If it does exist, the password is changed.

password
The plaintext password to be encrypted and stored in the DBM file. Used only with the -b flag.

-TDBTYPE
Type of DBM file (SDBM, GDBM, DB, or "default").
One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, GNU GDBM, and Berkeley/Sleepycat DB 2/3/4. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by `filename` is the same format that `htdbm` expects to see. `htdbm` currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.
htdbm returns a zero status ("true") if the username and password have been successfully added or updated in the DBM File. htdbm returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn't match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the Restrictions section), and 7 if the file is not a valid DBM password file.
htdbm /usr/local/etc/apache/.htdbm-users jsmith

Adds or modifies the password for user jsmith. The user is prompted for the password. If executed on a Windows system, the password will be encrypted using the modified Apache MD5 algorithm; otherwise, the system's crypt() routine will be used. If the file does not exist, htdbm will do nothing except return an error.

htdbm -c /home/doe/public_html/.htdbm jane

Creates a new file and stores a record in it for user jane. The user is prompted for the password. If the file exists and cannot be read, or cannot be written, it is not altered and htdbm will display a message and return an error status.

htdbm -mb /usr/web/.htdbm-all jones Pwd4Steve

Encrypts the password from the command line (Pwd4Steve) using the MD5 algorithm, and stores it in the specified file.
Web password files such as those managed by ht dbm should not be within the Web server's URI space -- that is, they should not be fetchable with a browser.

The use of the -b option is discouraged, since when it is used the unencrypted password appears on the command line.
On the Windows and MPE platforms, passwords encrypted with htdbm are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

The MD5 algorithm used by htdbm is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Usernames are limited to 255 bytes and may not include the character :.
htdigest is used to create and update the flat-files used to store usernames, realm and password for digest authentication of HTTP users. Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htdigest.

This manual page only lists the command line arguments. For details of the directives necessary to configure digest authentication in httpd see the Apache manual, which is part of the Apache distribution or can be found at http://httpd.apache.org/.

See also

httpd
mod_auth_digest
htdigest [ -c ] passwdfile realm username
-c
Create the *passwdfile*. If *passwdfile* already exists, it is deleted first.

**passwdfile**
Name of the file to contain the username, realm and password. If -c is given, this file is created if it does not already exist, or deleted and recreated if it does exist.

**realm**
The realm name to which the user name belongs. See [http://tools.ietf.org/html/rfc2617#section-3.2.1](http://tools.ietf.org/html/rfc2617#section-3.2.1) for more details.

**username**
The user name to create or update in *passwdfile*. If *username* does not exist is this file, an entry is added. If it does exist, the password is changed.
This program is not safe as a setuid executable. Do \textit{not} make it setuid.
htpasswd - Manage user files for basic authentication

htpasswd is used to create and update the flat-files used to store usernames and password for basic authentication of HTTP users. If htpasswd cannot access a file, such as not being able to write to the output file or not being able to read the file in order to update it, it returns an error status and makes no changes.

Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htpasswd. This program can only manage usernames and passwords stored in a flat-file. It can encrypt and display password information for use in other types of data stores, though. To use a DBM database see dbmmanage.

htpasswd encrypts passwords using either a version of MD5 modified for Apache, or the system's crypt() routine. Files managed by htpasswd may contain both types of passwords; some user records may have MD5-encrypted passwords while others in the same file may have passwords encrypted with crypt().

This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in httpd see the Apache manual, which is part of the Apache distribution or can be found at http://httpd.apache.org/.

See also

httpd
The scripts in support/SHA1 which come with the distribution.
htpasswd [ -c ] [ -m ] [ -D ] passwdfile username

htpasswd -b [ -c ] [ -m | -d | -p | -s ] [ -D ] passwdfile username password

htpasswd -n [ -m | -d | -s | -p ] username

htpasswd -nb [ -m | -d | -s | -p ] username password
-b
Use batch mode; *i.e.*, get the password from the command line rather than prompting for it. This option should be used with extreme care, since **the password is clearly visible** on the command line.

-c
Create the `passwdfile`. If `passwdfile` already exists, it is rewritten and truncated. This option cannot be combined with the `-n` option.

-n
Display the results on standard output rather than updating a file. This is useful for generating password records acceptable to Apache for inclusion in non-text data stores. This option changes the syntax of the command line, since the `passwdfile` argument (usually the first one) is omitted. It cannot be combined with the `-c` option.

-m
Use MD5 encryption for passwords. This is the default (since version 2.2.18).

-d
Use `crypt()` encryption for passwords. This is not supported by the `httpd` server on Windows and Netware and TPF. This algorithm limits the password length to 8 characters. This algorithm is **insecure** by today's standards. It used to be the default algorithm until version 2.2.17.

-s
Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (ldif).

-p
Use plaintext passwords. Though htpasswd will support creation on all platforms, the httpd daemon will only accept plain text passwords on Windows, Netware and TPF.

-D
Delete user. If the username exists in the specified htpasswd file, it will be deleted.

**passwdfile**
Name of the file to contain the user name and password. If -c is given, this file is created if it does not already exist, or rewritten and truncated if it does exist.

**username**
The username to create or update in passwdfile. If username does not exist in this file, an entry is added. If it does exist, the password is changed.

**password**
The plaintext password to be encrypted and stored in the file. Only used with the -b flag.
htpasswd returns a zero status ("true") if the username and password have been successfully added or updated in the passwdfile. htpasswd returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn’t match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the Restrictions section), and 7 if the file is not a valid password file.
htpasswd /usr/local/etc/apache/.htpasswd-users jsmith

Adds or modifies the password for user jsmith. The user is prompted for the password. The password will be encrypted using the modified Apache MD5 algorithm. If the file does not exist, htpasswd will do nothing except return an error.

htpasswd -c /home/doe/public_html/.htpasswd jane

Creates a new file and stores a record in it for user jane. The user is prompted for the password. If the file exists and cannot be read, or cannot be written, it is not altered and htpasswd will display a message and return an error status.

htpasswd -db /usr/web/.htpasswd-all jones Pwd4Steve

Encrypts the password from the command line (Pwd4Steve) using the crypt() algorithm, and stores it in the specified file.
Web password files such as those managed by htpasswd should not be within the Web server's URI space -- that is, they should not be fetchable with a browser.

This program is not safe as a setuid executable. Do not make it setuid.

The use of the -b option is discouraged, since when it is used the unencrypted password appears on the command line.

When using the crypt() algorithm, note that only the first 8 characters of the password are used to form the password. If the supplied password is longer, the extra characters will be silently discarded.

The SHA encryption format does not use salting: for a given password, there is only one encrypted representation. The crypt() and MD5 formats permute the representation by prepending a random salt string, to make dictionary attacks against the passwords more difficult.
On the Windows and MPE platforms, passwords encrypted with `htpasswd` are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

The MD5 algorithm used by `htpasswd` is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Usernames are limited to 255 bytes and may not include the character `:`.
logresolve - Resolve IP-addresses to hostnames in Apache log files

logresolve is a post-processing program to resolve IP-addresses in Apache's access logfiles. To minimize impact on your nameserver, logresolve has its very own internal hash-table cache. This means that each IP number will only be looked up the first time it is found in the log file.

Takes an Apache log file on standard input. The IP addresses must be the first thing on each line and must be separated from the remainder of the line by a space.
logresolve [ -s filename ] [ -c ] < access_log > access_log.new
-s filename
   Specifies a filename to record statistics.

-c
   This causes log resolve to apply some DNS checks: after finding
   the hostname from the IP address, it looks up the IP
   addresses for the hostname and checks that one of these
   matches the original address.
rotatelog - Piped logging program to rotate Apache logs

rotatelog is a simple program for use in conjunction with Apache's piped logfile feature. It supports rotation based on a time interval or maximum size of the log.
rotatelogs [ -l ] [ -f ] logfile
rotationtime|filesizeM [ offset ]
-l
Causes the use of local time rather than GMT as the base for the interval or for strftime(3) formatting with size-based rotation. Note that using -l in an environment which changes the GMT offset (such as for BST or DST) can lead to unpredictable results!

-f
Causes the logfile to be opened immediately, as soon as rotatelogs starts, instead of waiting for the first logfile entry to be read (for non-busy sites, there may be a substantial delay between when the server is started and when the first request is handled, meaning that the associated logfile does not "exist" until then, which causes problems from some automated logging tools). Available in version 2.2.9 and later.

logfile
The path plus basename of the logfile. If logfile includes any '%' characters, it is treated as a format string for strftime(3). Otherwise, the suffix .nnnnnnnnnn is automatically added and is the time in seconds. Both formats compute the start time from the beginning of the current period. For example, if a rotation time of 86400 is specified, the hour, minute, and second fields created from the strftime(3) format will all be zero, referring to the beginning of the current 24-hour period (midnight).

When using strftime(3) filename formatting, be sure the log file format has enough granularity to produce a different file name each time the logs are rotated. Otherwise rotation will overwrite the same file instead of starting a new one. For example, if logfile was /var/logs/errorlog.%Y-%m-%d with log rotation at 5 megabytes, but 5 megabytes was reached twice in the same day, the same log file name would
be produced and log rotation would keep writing to the same file.

**rotationtime**

The time between log file rotations in seconds. The rotation occurs at the beginning of this interval. For example, if the rotation time is 3600, the log file will be rotated at the beginning of every hour; if the rotation time is 86400, the log file will be rotated every night at midnight. (If no data is logged during an interval, no file will be created.)

**filesizeM**

The maximum file size in megabytes followed by the letter M to specify size rather than time.

**offset**

The number of minutes offset from UTC. If omitted, zero is assumed and UTC is used. For example, to use local time in the zone UTC -5 hours, specify a value of -300 for this argument. In most cases, -1 should be used instead of specifying an offset.
Examples

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

This creates the files /var/logs/logfile.nnnn where nnnn is the system time at which the log nominally starts (this time will always be a multiple of the rotation time, so you can synchronize cron scripts with it). At the end of each rotation time (here after 24 hours) a new log is started.

CustomLog "|bin/rotatelogs -l /var/logs/logfile.%Y.%m.%d 86400" common

This creates the files /var/logs/logfile.yyyy.mm.dd where yyyy is the year, mm is the month, and dd is the day of the month. Logging will switch to a new file every day at midnight, local time.

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

This configuration will rotate the logfile whenever it reaches a size of 5 megabytes.

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

This configuration will rotate the error logfile whenever it reaches a size of 5 megabytes, and the suffix to the logfile name will be created of the form errorlog.YYYY-mm-dd-HH_MM_SS.
The following logfile format string substitutions should be supported by all `strftime(3)` implementations, see the `strftime(3)` man page for library-specific extensions.

<table>
<thead>
<tr>
<th>Format Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%A</td>
<td>full weekday name (localized)</td>
</tr>
<tr>
<td>%a</td>
<td>3-character weekday name (localized)</td>
</tr>
<tr>
<td>%B</td>
<td>full month name (localized)</td>
</tr>
<tr>
<td>%b</td>
<td>3-character month name (localized)</td>
</tr>
<tr>
<td>%c</td>
<td>date and time (localized)</td>
</tr>
<tr>
<td>%d</td>
<td>2-digit day of month</td>
</tr>
<tr>
<td>%H</td>
<td>2-digit hour (24 hour clock)</td>
</tr>
<tr>
<td>%I</td>
<td>2-digit hour (12 hour clock)</td>
</tr>
<tr>
<td>%j</td>
<td>3-digit day of year</td>
</tr>
<tr>
<td>%M</td>
<td>2-digit minute</td>
</tr>
<tr>
<td>%m</td>
<td>2-digit month</td>
</tr>
<tr>
<td>%p</td>
<td>am/pm of 12 hour clock (localized)</td>
</tr>
<tr>
<td>%S</td>
<td>2-digit second</td>
</tr>
<tr>
<td>%U</td>
<td>2-digit week of year (Sunday first day of week)</td>
</tr>
<tr>
<td>%W</td>
<td>2-digit week of year (Monday first day of week)</td>
</tr>
<tr>
<td>%W</td>
<td>1-digit weekday (Sunday first day of week)</td>
</tr>
<tr>
<td>%X</td>
<td>time (localized)</td>
</tr>
<tr>
<td>%x</td>
<td>date (localized)</td>
</tr>
<tr>
<td>%y</td>
<td>4-digit year</td>
</tr>
<tr>
<td>%Y</td>
<td>2-digit year</td>
</tr>
<tr>
<td>%Z</td>
<td>time zone name</td>
</tr>
<tr>
<td>%%</td>
<td>literal `%'</td>
</tr>
</tbody>
</table>
Other Programs

The following programs are simple support programs included with the Apache HTTP Server which do not have their own manual pages. They are not installed automatically. You can find them after the configuration process in the support/ directory.
This perl script is designed to be run at a frequent interval by something like cron. It connects to the server and downloads the status information. It reformats the information to a single line and logs it to a file. Adjust the variables at the top of the script to specify the location of the resulting logfile.
This perl script will take a combined Web server access log file and break its contents into separate files. It assumes that the first field of each line is the virtual host identity (put there by "%v"), and that the logfiles should be named that + ".log" in the current directory.

The combined log file is read from stdin. Records read will be appended to any existing log files.

```
split-logfile < access_log
```
Relevant Standards

This page documents all the relevant standards that the Apache HTTP Server follows, along with brief descriptions.

In addition to the information listed below, the following resources should be consulted:

- [http://purl.org/NET/http-errata](http://purl.org/NET/http-errata) - HTTP/1.1 Specification Errata

Notice

This document is not yet complete.
Regardless of what modules are compiled and used, Apache as a basic web server complies with the following IETF recommendations:

**RFC 1945 (Informational)**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. This documents HTTP/1.0.

**RFC 2616 (Standards Track)**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This documents HTTP/1.1.

**RFC 2396 (Standards Track)**

A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.
Regarding the Hypertext Markup Language, Apache complies with the following IETF and W3C recommendations:

**RFC 2854 (Informational)**

This document summarizes the history of HTML development, and defines the "text/html" MIME type by pointing to the relevant W3C recommendations.

**HTML 4.01 Specification (Errata)**

This specification defines the HyperText Markup Language (HTML), the publishing language of the World Wide Web. This specification defines HTML 4.01, which is a subversion of HTML 4.

**HTML 3.2 Reference Specification**

The HyperText Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents.

**XHTML 1.1 - Module-based XHTML (Errata)**

This Recommendation defines a new XHTML document type that is based upon the module framework and modules defined in Modularization of XHTML.

**XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) (Errata)**

This specification defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4.
Concerning the different methods of authentication, Apache follows the following IETF recommendations:

**RFC 2617 (Draft standard)**

"HTTP/1.0", includes the specification for a Basic Access Authentication scheme.
The following links document ISO and other language and country code information:

**ISO 639-2**
ISO 639 provides two sets of language codes, one as a two-letter code set (639-1) and another as a three-letter code set (this part of ISO 639) for the representation of names of languages.

**ISO 3166-1**
These pages document the country names (official short names in English) in alphabetical order as given in ISO 3166-1 and the corresponding ISO 3166-1-alpha-2 code elements.

**BCP 47 (Best Current Practice), RFC 3066**
This document describes a language tag for use in cases where it is desired to indicate the language used in an information object, how to register values for use in this language tag, and a construct for matching such language tags.

**RFC 3282 (Standards Track)**
This document defines a "Content-language:" header, for use in cases where one desires to indicate the language of something that has RFC 822-like headers, like MIME body parts or Web documents, and an "Accept-Language:" header for use in cases where one wishes to indicate one's preferences with regard to language.
Apache

MPM
  "MPM" Apache MPM

Base
  "Base"

Extension
  "Extension"

Experimental
  "Experimental" Apache

External
  "External" Apache (""")
Apache
**URL**

http://www.example.com/path/to/file.html

() Uniform Resource Loca: **URL-path**

/path/to/file.html url

**file-path**

/usr/local/apache/htdocs/path/to/file.html

file-path ServerRoot

**directory-path**

/usr/local/apache/htdocs/path/to/

**filename**

file.html

**regex**

Perl regex

**extension**

filename Apache

filename file.html.en

extension

**MIME-type**

text/html

**env-variable**

Apache
Apache
(httpd.conf, srm.conf, access.conf

<VirtualHost>
  <Directory>
    .htaccess
  </Directory>
</VirtualHost>

<VirtualHost>
  <Directory>
    <Location>
      <Files>
        .htaccess
      </Files>
    </Location>
  </Directory>
</VirtualHost>

.htaccess
  .htaccess

httpd.conf  .htaccess
  <VirtualHost>
  <Location>
    <Files>
      .htaccess
    </Files>
  </Location>
</VirtualHost>

(: Boolean OR

<Directory>
</Directory>
.htaccess

AllowOverride ()
Apache

Core
   "Core" Apache

MPM
   "MPM"

Base
   "Base"

Extension
   "Extension" Apache

Experimental
   "Experimental" Apache
Apache

Apache HTTP

Core
Listen
AcceptFilter protocol accept_filter

Core
core
2.1.5

Listen OS
Accept Filter Linux TCP_DEFER_ACCEPT

FreeBSD:
AcceptFilter http httpready
AcceptFilter https dataready

httpready Accept Filter HTTP HTTPS accf_data(9)

Linux:
AcceptFilter http data
AcceptFilter https data

Linux TCP_DEFER_ACCEPT HTTP TCP_DEFER_ACCEPT Linux man

none Accept Filter

AcceptFilter nntp none
AcceptPathInfo: On|Off|Default

AcceptPathInfo: Default

PathInfo: .htaccess

FileInfo:

Core:

Apache 2.0.30

(test/

/test/here.html
/test/nothere.html/more

AcceptPathInfo: Off

On

Default

PATH_INFO

AcceptPathInfo: PATH_INFO

PATH_INFO

<Files "mypaths.shtml">
  Options +Includes
  SetOutputFilter INCLUDES
  AcceptPathInfo On
</Files>
AccessFileName filename [filename] ...

AccessFileName .htaccess
.
Core
core

AccessFileName .acl

/usr/
/usr/.acl,/usr/local/.acl,/usr/local/web/.acl

<Directory />
  AllowOverride None
</Directory>

- AllowOverride
- .htaccess
AddDefaultCharset On|Off|charset
AddDefaultCharset Off
,.,.htaccess
FileInfo
Core
core

AddDefaultCharset

AddDefaultCharset utf-8

AddDefaultCharset
AddOutputFilterByType "filter[;filter...]
  MIME-type [MIME-type] ...
  , , .htaccess
  FileInfo

  Core
  core
  Apache 2.0.33 2.1

MIME

mod_deflate DEFLATE text/html text/plain
()

AddOutputFilterByType DEFLATE text/html text/plain

<Location /cgi-bin/>
  Options Includes
  AddOutputFilterByType INCLUDES;DEFLATE text/html
</Location>

AddOutputFilterByType
DefaultType

DefaultType
- AddOutputFilter
- SetOutputFilter
AllowEncodedSlashes	On
URL 404 (Not found)

AllowEncodedSlashes Off

Path_Arguments

PATH_INFO

%2F ()
%5C URL

- [AcceptPathInfo](#)
AllowOverride All|None|directive-type

[directive-type] ...

AllowOverride All

Core
core

( AccessFileName ) .htaccess

<Directory>
AllowOverride <Directory> <Location>
<DirectoryMatch> <Files>

None .htaccess

All .htaccess
directive-type

AuthConfig

(AuthDBMGroupFile, AuthGroupFile, AuthName, AuthType, AuthUserFile, Require)

FileInfo

(DefaultType, ErrorDocument, 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_actions  Action

Indexes
(AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadmeName)

Limit
(Allow Options[=Option,...])

: AllowOverride AuthConfig Indexes

AuthConfig  Indexes

- AccessFileName
- .htaccess
HTTP (: realm)
: AuthName auth-domain
, .htaccess
: AuthGroupFile
: Core
: core

( : realm)
Require AuthUserFile AuthGroupFile

AuthName "Top Secret"

AuthName
AuthType Basic|Digest, .htaccess
AuthConfig Core
core

(mod_auth_digest)

AuthName Require
mod_authz_user
Apache CGI

.foo .foo CGI FOO
<table>
<thead>
<tr>
<th>Setting</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-MD5 HTTP</td>
<td>On</td>
</tr>
<tr>
<td>ContentDigest On</td>
<td>Off</td>
</tr>
<tr>
<td>Core</td>
<td>On</td>
</tr>
</tbody>
</table>

RFC1864  RFC2616  Content-MD5:

```
Content-MD5: AuLb7Dp1rqtRtxz2m9kRpA==
```
MIME

DefaultType MIME-type|none
DefaultType text/plain

., .htaccess
FileInfo
Core
core

MIME

(SHOULD)

DefaultType image/gif

.gif

GIF (proxy)

DefaultType None

DefaultType None httpd-2.2.7

ForceType MIME
path Unix
  /home/user/public_html  <Directory
/*/*.public_html>  <Directory
/home/*/public_html>  :

<Directory /usr/local/httpd/htdocs>
  Options IndexesFollowSymLinks
</Directory>

directory-path: Apache  <Directory

~  :

<Directory - "^/www/*./[0-9]{3}"

/www/  3
()

<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/.htaccess
  FileInfo

```
<Directory ~\tabc$>
    # ... directives here ...
</Directory>
```

```
<Directory> .htaccess
/home/abc/public_html/abc <Directory>

Apache  <Directory />  Allow from All
URL Apache

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

```
httpd.conf  <Directory>
<LimitExcept>
```

```
<Directory>
  <DirectoryMatch regex> ...
  <DirectoryMatch>
  ,
  Core
  core
</Directory>

<DirectoryMatch "^/www/(.+)?[0-9]{3}">
/WWW/ 3

  <Directory>  <Directory>
  
</DirectoryMatch>
DocumentRoot `directory-path`

DocumentRoot `/usr/local/apache/htdocs`

Core

core

**httpd**

<table>
<thead>
<tr>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http://www.my.host.com/index.html</code></td>
</tr>
<tr>
<td><code>/usr/web/index.html</code></td>
</tr>
</tbody>
</table>

**URL**
EnableMMAP On|Off

EnableMMAP On

.,.\.htaccess

FileInfo

Core
core

NFS

DocumentRoot

httpd

<Directory "/path-to-nfs-files">
  EnableMMAP Off
</Directory>
sendfile

EnableSendfile On|Off
EnableSendfile On
.
htaccess
FileInfo
Core
core
2.0.44

httpd

sendfile read send

- sendfile
- Linux sendfile IPv6 TCP-checksum
- Itanium Linux sendfile 2GB
- DocumentRoot (NFS SMB)

EnableSendfile Off

NFS SMB

<Directory "/path-to-nfs-files">
    EnableSendfile Off
</Directory>
ErrorDocument 500 http://foo.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.pl
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 "Sorry can't allow you access today"

default Apache

ErrorDocument Apache

ErrorDocument 404 /cgi-bin/bad_urls.pl

DIRECTORY /web/docs
  ErrorDocument 404 default
</DIRECTORY>
URL

Apache
  ErrorDocument 401 URL 401
  ErrorDocument 401

Microsoft Internet Explorer (MSIE)
MSIE Microsoft

2.0

Error
ErrorLog

ErrorLog

ErrorLog /var/log/httpd/error_log

file-path ()

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

syslog

syslog:facility

syslogd(8)

syslog(1)

ErrorLog syslog:user

Unix
- LogLevel
- Apache
<table>
<thead>
<tr>
<th>FileETag</th>
<th>inode, (mtime)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INode</td>
<td>inode</td>
</tr>
<tr>
<td>MTime</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>None</th>
<th>ETag</th>
</tr>
</thead>
<tbody>
<tr>
<td>INode, MTime, Size</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FileETag</th>
<th>INode</th>
<th>MTime</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebDAV</td>
<td>mod_dav_fs</td>
<td>Directory</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td><strong>mod_dav_fs</strong></td>
<td>INode MTime Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FileETag</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**mod_dav_fs** is a plugin for the Apache web server that implements the WebDAV protocol.
<Files filename> ...
</Files>
,,,.htaccess
All
Core
core

<Files>
  ()
.htaccess
</Files>  <Location>
<Directory>

filename

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

<Directory>  <Location>  <Files>  .

.
<FilesMatch regex> ... </FilesMatch>
..., .htaccess
All
Core
core

<FilesMatch> <Files>

<FilesMatch "\.(gif|jpe?g|png)$">
<table>
<thead>
<tr>
<th>MIME Type</th>
<th>ForceType</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>MIME-type</td>
</tr>
<tr>
<td>.htaccess</td>
<td>FileInfo</td>
</tr>
<tr>
<td>Core</td>
<td>Core</td>
</tr>
<tr>
<td>Apache 2.0</td>
<td>core</td>
</tr>
</tbody>
</table>

```plaintext
# .htaccess

<Directory>
  ForceType image/gif
</Directory>

DefaultType

None  ForceType :

# force all files to be image/gif:
<Location /images>
  ForceType image/gif
</Location>

# but normal mime-type associations here:
<Location /images/mixed>
  ForceType None
</Location>
```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GprofDir</strong></td>
<td>Directory to write gmon.out profiling data to.</td>
</tr>
<tr>
<td></td>
<td>/tmp/gprof/</td>
</tr>
<tr>
<td></td>
<td>,</td>
</tr>
<tr>
<td><strong>Core</strong></td>
<td>Core</td>
</tr>
<tr>
<td></td>
<td>core</td>
</tr>
<tr>
<td>IP</td>
<td>DNS</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DNS
IP

`mod_authz_host`
Double 2
REMOTE_HOST

bin
Modify restrictions on HTTP Request Messages

HttpProtocolOptions [Strict|Unsafe]
  [RegisteredMethods|LenientMethods]
  [Allow0.9|Require1.0]

HttpProtocolOptions Strict LenientMethods
  Allow0.9

, Core
  core

  2.2.32 or 2.4.24 and later
<IfDefine test>...<IfDefine>
    test test
<IfDefine>
    test :
    
    • parameter-name
    • !parameter-name

    parameter-name

    parameter-name httpd httpd -Dpa
<IfDefine>

httpd -DReverseProxy ...

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

<IfModule> test

module
!module

module Apache ( STANDARD20

<IfModule>
Include file-path|directory-path

Core: core
Core: 2.0.41

ServerRoot:

httpd

ServerRoot:

Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf

apachectl configtest:

root@host# apachectl configtest
Processing config file: /usr/local/apache2/conf/ssl.conf
Processing config file: /usr/local/apache2/conf/vhosts/vhost1.conf
Processing config file: /usr/local/apache2/conf/vhosts/vhost2.conf
Syntax OK
• **apachectl**
HTTP
KeepAlive On|Off
KeepAlive On
Core
core

HTTP/1.0 Keep-Alive HTTP/1.1
TCP
HTML 50%

HTTP/1.0
Keep-Alive
CGI SSI
HTTP/1.1

- MaxKeepAliveRequests
KeepAliveTimeout seconds
KeepAliveTimeout 5

Apache

KeepAliveTimeout
<Limit method [method] ... > ... </Limit>

..., .htaccess

All

Core

core

<Limit>

HTTP

DELETE

<Limit POST PUT DELETE>

Require valid-user

</Limit>

GET, POST, PUT, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK.

GET

HEAD

TRACE

<Limit>

<LimitExcept>
<LimitExcept method [method] ... > ...</LimitExcept>

<LimitExcept . . . .htaccess

All

Core

core

<LimitExcept>  </LimitExcept>  HTTP

<Limit>  

</LimitExcept>

<LimitExcept POST GET>

Require valid-user
</LimitExcept>
LimitInternalRecursion number [number]

LimitInternalRecursion 10

Core
core

Apache 2.0.47

Action mod_dir DirectoryIndex

LimitInternalRecursion

LimitInternalRecursion 5
HTTP
LimitRequestBody bytes
LimitRequestBody 0
.,.htaccess
All
Core
core

bytes 0 () 2147483647 (2GB)

LimitRequestBody

100K

LimitRequestBody 102400
LimitRequestFields number

LimitRequestFields 100

Core

core

number 0 () 32767
DEFAULT_LIMIT_REQUEST_FIELDS (100)

LimitRequestBody

HTTP

HTTP

HTTP

LimitRequestFields 50
LimitRequestFieldSize: HTTP

LimitRequestFieldSize 8190

Core

core
HTTP

LimitRequestLine bytes

LimitRequestLine 8190

Core

core

HTTP

bytes

LimitRequestLine HTTP

LimitRequestLine URI

LimitRequestLine 4094
XML

LimitXMLRequestBody bytes

LimitXMLRequestBody 1000000

..., .htaccess

All

Core

core

XML ()

LimitXMLRequestBody ()
<table>
<thead>
<tr>
<th>Location</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Location URL-path&gt; ... &lt;/Location&gt;</td>
<td>,</td>
</tr>
<tr>
<td>Core</td>
<td>core</td>
</tr>
</tbody>
</table>

**<Location>**

**<Files>**

**<Location>**

**<Location>**

<table>
<thead>
<tr>
<th>Location</th>
<th>URL</th>
<th>http scheme://servername/path</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td>/path/</td>
<td>*</td>
</tr>
<tr>
<td>~</td>
<td>:</td>
<td></td>
</tr>
</tbody>
</table>

| Location ~ "/(extra|special)/data" |
|-----------------------------------|

<table>
<thead>
<tr>
<th>URL</th>
<th>/extra/data</th>
<th>/special/data</th>
</tr>
</thead>
</table>

**<LocationMatch>**

**<Location>**

**<Location>**

**<Location>**

**SetHandler**

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

<!--

URL

<LocationMatch>
</Location>

<LocationMatch ^/abc>
</Location>

URL

<LocationMatch ^/abc> /abc URL
</LocationMatch>

<Location> proxy
</Location>

/abc//def

-->

..
<LocationMatch>
  <LocationMatch regex> ... </LocationMatch>
</LocationMatch>

Core
  core

<LocationMatch>
  <Location>
    URL
    
    <LocationMatch "/(extra|special)/data">
      URL  /extra/data  /special/data
    </LocationMatch>
  </Location>
</LocationMatch>
LogLevel:

<table>
<thead>
<tr>
<th>LogLevel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emerg</td>
<td>Child cannot open lock file.Exiting ()</td>
</tr>
<tr>
<td>alert</td>
<td>getpwuid: couldn't determine user name from uid (getpwuid: UID )</td>
</tr>
<tr>
<td>crit</td>
<td>socket: Failed to get a socket, exiting child (socket: )</td>
</tr>
<tr>
<td>error</td>
<td>Premature end of script headers ()</td>
</tr>
<tr>
<td>warn</td>
<td>child process 1234 did not exit, sending another SIGHUP ( 1234 SIGHUP )</td>
</tr>
<tr>
<td>notice</td>
<td>httpd: caught SIGBUS, attempting to dump core in ... (httpd: SIGBUS ... )</td>
</tr>
<tr>
<td>info</td>
<td>&quot;Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)...&quot; (StartServers Min/MaxSpareServers )</td>
</tr>
<tr>
<td>debug</td>
<td>&quot;Opening config file ...&quot; (....)</td>
</tr>
</tbody>
</table>

**LogLevel** ( **ErrorLog** )
LogLevel notice

notice
MaxKeepAliveRequests number
MaxKeepAliveRequests 100
, Core
core

MaxKeepAliveRequests 500
Number of ranges allowed before returning the complete resource

MaxRanges default | unlimited | none | number-of-ranges

MaxRanges 200

Available in Apache HTTP Server 2.2.21 and later
MergeTrailers

Determines whether trailers are merged into headers

MergeTrailers [on|off]

MergeTrailers off

Core

core

2.2.28 and later
<table>
<thead>
<tr>
<th>addr</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>111.22.33.44</td>
<td></td>
</tr>
<tr>
<td><em>default</em></td>
<td></td>
</tr>
<tr>
<td>111.22.33.44:8080</td>
<td></td>
</tr>
<tr>
<td>[2001:db8::a00:20ff:fe17:ceea]:8080</td>
<td>*</td>
</tr>
</tbody>
</table>

```bash
NameVirtualHost 111.22.33.44
NameVirtualHost _default_
NameVirtualHost 111.22.33.44:8080
NameVirtualHost [2001:db8::a00:20ff:fe17:ceea]:8080
```
<VirtualHost>
<VirtualHost> NameVirtualHost

NameVirtualHost 1.2.3.4
<VirtualHost 1.2.3.4>
# ...
</VirtualHost>
</VirtualHost>
Options

| option | None | 1 |

**All**
- MultiViews

**ExecCGI**
- mod_cgi CGI

**FollowSymLinks**

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

**Includes**
- mod_include SSI

**IncludesNOEXEC**
- SSI #exec #exec CGI
- virtual ScriptAlias CGI

**Indexes**
<Location>
    symlink
</Location>

Options +

Options + -

+ -:

<Directory /web/docs>
    Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
    Options Includes
</Directory>

/web/docs/spec Includes ;

-:

<Directory /web/docs>
    Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
  Options +Includes -Indexes
</Directory>

/web/docs/spec  FollowSymLinks  Includes

-IncludesNOEXEC  -Includes  SSI

All
<table>
<thead>
<tr>
<th>Protocol for a listening socket</th>
<th>Protocol protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>core</td>
</tr>
<tr>
<td>Available in Apache 2.1.5 and later. On Windows, from Apache 2.3.3 and later.</td>
<td></td>
</tr>
</tbody>
</table>

- `AcceptFilter`
- `Listen`
Register non-standard HTTP methods

RegisterHttpMethod method [method [...]]

Core
core
Require user \textit{userid} [\textit{userid}] ...

Require group \textit{group-name} [\textit{group-name}] ...

Require valid-user

\begin{verbatim}
require mod_authz_owner
mod_authnz_ldap

Require AuthName AuthType () AuthUserFile AuthGroupFile

AuthType Basic
AuthName "Restricted Resource"
AuthUserFile /web/users
AuthGroupFile /web/groups
Require group admin
\end{verbatim}

Require Allow Deny
Satisfy

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

- Satisfy
- mod_authz_host
<table>
<thead>
<tr>
<th>RLimitCPU</th>
<th>seconds</th>
<th>max</th>
<th>seconds</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>.htaccess</td>
<td></td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>core</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Apache

Apache fork

CPU

- **RLimitMEM**
- **RLimitNPROC**
| Apache | RLimitMEM bytes|max | bytes|max |
|--------|-------------------|--------|
|        | , , .htaccess      | All    |
|        | Core              | core   |

- **RLimitCPU**
- **RLimitNPROC**
| RLimitNPROC | number | max | [number | max] |
|------------|--------|-----|----------|

All .htaccess

Core

core

Apache

Apache fork

Apache fork

CGI ID

- RLimitMEM
- RLimitCPU
Apache CGI

```perl
#!C:/Perl/bin/perl.exe

perl PATH:

#!perl

ScriptInterpreterSource Registry (Windows HKEY_CLASSES_ROOT Shell\ExecCGI\Command Apache Script .pl perl

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

ScriptInterpreterSource Registry ScriptAlias
```
ServerAdmin

mailto: www-admin@foo.example.com
ServerAlias hostname [hostname] ...

Core

core

ServerAlias

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

- Apache
<table>
<thead>
<tr>
<th>ServerName</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple.example.com DNS</td>
<td><a href="http://www.example.com">www.example.com</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ServerName</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.example.com:80">www.example.com:80</a></td>
<td></td>
</tr>
</tbody>
</table>

```<VirtualHost>```

<table>
<thead>
<tr>
<th>SSL</th>
<th>SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL (mod_dir)</td>
<td>UseCanonicalPhysicalPort</td>
</tr>
</tbody>
</table>

- **DNS**
- **Apache**
- **UseCanonicalName**
- **UseCanonicalPhysicalPort**
- **NameVirtualHost**
- **ServerAlias**
<table>
<thead>
<tr>
<th>ServerPath</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL-path</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>core</td>
</tr>
</tbody>
</table>

- Apache
ServerRoot directory-path
ServerRoot /usr/local/apache

Core
core

ServerRoot /home/httpd

- httpd -d
- ServerRoot
<table>
<thead>
<tr>
<th>ServerSignature</th>
<th>On</th>
<th>Off</th>
<th>EMail</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerSignature</td>
<td>Off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.htaccess</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>core</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ServerSignature

ServerName

EMail ServerAdmin "mailto:"

2.0.44 ServerSignature

- ServerTokens
<table>
<thead>
<tr>
<th>ServerHTTP</th>
<th>ServerTokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>Minor</td>
</tr>
<tr>
<td>ServerTokens Full</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td></td>
</tr>
<tr>
<td>core</td>
<td></td>
</tr>
</tbody>
</table>

**Server OS**

**ServerTokens ProductOnly**

```
(): Server: Apache
```

**ServerTokens Major**

Server sends (e.g.): Server: Apache/2

**ServerTokens Minor**

Server sends (e.g.): Server: Apache/2.0

**ServerTokens Minimal**

```
(): 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 "handler-name" None

, ., .htaccess

FileInfo

Core
core
Apache 2.0 core

.htaccess <Directory> <Location>
name

SetHandler imap-file

: URL http://servername/status

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

None SetHandler

: SetHandler (/) URL

  • AddHandler
SetInputFilter

POST

SetInputFilter  filter[;filter...]

, , , .htaccess

FileInfo

Core

core

SetInputFilter  POST
SetOutputFilter

/www/data/          SSI

<Directory /www/data/>
  SetOutputFilter INCLUDES
</Directory>
<table>
<thead>
<tr>
<th>Enable or disable the suEXEC feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suexec On</td>
</tr>
</tbody>
</table>

On if suexec binary exists with proper owner and mode, Off otherwise

- Core
- core
- Available in Apache httpd 2.2.18 and later
TimeOut:

TimeOut seconds:

TimeOut 300:

, Core:

core:

TimeOut:

1. GET
2. POST PUT TCP
3. TCP ACK

Apache 1.2 1200
TraceEnable [on|off|extended]
TraceEnable on
Core apache 1.3.34, 2.0.55
mod_proxy TRACE
RFC2616 TRACE TraceEnable off
405 ()
Transfer-Encoding: chunked
chunk proxy 64k
chunk +8k)
UseCanonicalName On/Off/Dns
UseCanonicalName Off

Core

core

Apache URL URL Use
ServerName ()
SERVER_NAME SERVER_PORT

UseCanonicalName Off
CGI SERVER_NAME SERVER_PORT

www
http://www.domain.com/splat/ 1 www.domain.com 1 --

UseCanonicalName Off Apache
http://www/splat/

UseCanonicalName DNS IP
Apache IP DNS URL

- UseCanonicalPhysicalPort
- ServerName
- Listen
<table>
<thead>
<tr>
<th>UseCanonicalPhysicalPort</th>
<th>On</th>
<th>Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseCanonicalPhysicalPort</td>
<td>Off</td>
<td></td>
</tr>
</tbody>
</table>

Core: core

Apache URL

<table>
<thead>
<tr>
<th>UseCanonicalPhysicalPort</th>
<th>On</th>
<th>URL</th>
<th>Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseCanonicalPhysicalPort</td>
<td>On</td>
<td></td>
<td>Off</td>
</tr>
</tbody>
</table>

UseCanonicalName:

- **ServerName**
- **Host:**
- **ServerName**

UseCanonicalPhysicalPort Off

UseCanonicalName:

- **UseCanonicalName**
- ServerName
- Listen
<VirtualHost>    </VirtualHost>

<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>

IPv6

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

IP alias )
<VirtualHost> Apache Listen IP Listen Apache listen

IP _default_
VirtualHost _default_
:port
)

<VirtualHost> ServerName "main"
ServerName

- Apache
- DNS_Apache
- Apache
Apache MPM

(MPM)

MPM
AcceptMutex

AcceptMutex default

AcceptMutex default

MPM

prefork, worker

AcceptMutex accept

Default

flock

LockFile flock(2)

fcntl

LockFile fcntl(2)

posixsem

POSIX

pthread

POSIX Threads (PThreads) POSIX

sysvsem

SySV

LogLev

pthread

AcceptCnt1 mutex

Solaris Apache mutex mutex

pthread_mutexattr_setrobust_np()
<table>
<thead>
<tr>
<th>ChrootDir</th>
<th>Directory for apache to run chroot(8) after startup.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ChrootDir /path/to/directory</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td>MPM</td>
<td>event, prefork, worker</td>
</tr>
<tr>
<td></td>
<td>Available in Apache 2.2.10 and later</td>
</tr>
<tr>
<td>MPM</td>
<td>beos, mpm_winnt, prefork, worker</td>
</tr>
</tbody>
</table>

Apache

**Linux**

<table>
<thead>
<tr>
<th>Apache root</th>
<th>Linux CoreDumpDirectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td></td>
</tr>
</tbody>
</table>
EnableExceptionHook On|Off
EnableExceptionHook Off

MPM
prefork, worker
2.0.49

--enable-exception-hook configure

mod_whokilledus n

Trawick EnableExceptionHook site
Graceful Shutdown Timeout: 

Graceful Shutdown Timeout: 0

MPM:

prefork, worker, event

2.2

Graceful Shutdown Timeout: "graceful-stop"
Group

Group unix-group

Group # -1

MPM

beos, mpmt_os2, prefork, worker

Apache 2.0

Group www-group

nobody

Group ( User)

<VirtualHost> Apache 2.0

SuexecUserGroup

Group beos mpmt_os2 MPM
Listen

<table>
<thead>
<tr>
<th>listen IP</th>
<th>Listen [IP-address:]portnumber [protocol]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPM</td>
<td>beos, mpm_netware, mpm_winnt, mpm_os2, prefork, worker, event</td>
</tr>
<tr>
<td>Apache 2.0 protocol 2.1.5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listen Apache</th>
<th>IP listen Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>listen 80</td>
<td>Listen 80</td>
</tr>
<tr>
<td>listen 8000</td>
<td>Listen 8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listen 192.170.2.1:80</th>
<th>Listen 192.170.2.5:8000</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>IPv6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Listen [2001:db8::a00:20ff:fea7:ccea]:80</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protocol 443</th>
<th>AcceptFilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol h1</td>
<td></td>
</tr>
</tbody>
</table>
Listen 192.170.2.1:8443 https

<table>
<thead>
<tr>
<th>IP</th>
<th>Listen</th>
</tr>
</thead>
</table>

- DNS
- Apache
ListenBacklog *backlog*

ListenBacklog 511

MPM

beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker

OS

OS OS
LockFile `filename`  
LockFile `logs/accept.lock`  

MPM
  * prefork, worker

**AcceptMutex**  
*fcntl*  
flock
logs  
NFS

/var/tmp
<table>
<thead>
<tr>
<th>MPM</th>
<th>MaxClients</th>
<th>ServerLimit</th>
</tr>
</thead>
<tbody>
<tr>
<td>beos, prefork, worker</td>
<td>16</td>
<td>25 (ThreadsPerChild)</td>
</tr>
</tbody>
</table>

MaxClients

(prefork) MaxClients

ServerLimit

(prefork) MaxClients

MPM

ServerLimit

MaxClients 16
MaxMemFree  KBytes
MaxMemFree  0

MPM
beos, mpm_netware, prefork, worker, mpm_winnt

MaxMemFree  free()
MaxRequestsPerChild

MaxRequestsPerChild 10000

MPM

mpm_netware, mpm_winnt, mpm_os2, prefork, worker

MaxRequestsPerChild

MaxRequestsPerChild 0

mpm_netware  mpm_winnt  0

KeepAlive
### MaxSpareThreads

<table>
<thead>
<tr>
<th>MPM</th>
<th>MaxSpareThreads</th>
<th>MPM</th>
<th>MaxSpareThreads</th>
</tr>
</thead>
<tbody>
<tr>
<td>worker</td>
<td>250</td>
<td>MPM</td>
<td>100</td>
</tr>
<tr>
<td>mpm_netware</td>
<td>100</td>
<td>beos</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>mpm_netware</td>
<td>mpmt_os2</td>
<td>10</td>
</tr>
</tbody>
</table>

### MinSpareThreads

- **Apache**
  - mpm_netware
  - worker

### ThreadsPerChild

- **Apache**
  - MinSpareThreads
  - StartServers
### MinSpareThreads

<table>
<thead>
<tr>
<th>MPM</th>
<th>MinSpareThreads</th>
</tr>
</thead>
<tbody>
<tr>
<td>beos, mpm_netware, mpmt_os2, worker</td>
<td>MPM worker MinSpareThreads 75</td>
</tr>
<tr>
<td>mpm_netware</td>
<td>MPM beos MinSpareThreads 10</td>
</tr>
<tr>
<td>beos</td>
<td>MPM mpmt_os2 MinSpareThreads 1</td>
</tr>
<tr>
<td>mpmt_os2</td>
<td>MPM mpm_netware MinSpareThreads 5</td>
</tr>
</tbody>
</table>

- **MaxSpareThreads**
- **StartServers**
### PidFile

<table>
<thead>
<tr>
<th>ID</th>
<th>PidFile filename</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>logs/httpd.pid</td>
</tr>
</tbody>
</table>

### MPM

- beos
- mpm_winnt
- mpmt_os2
- prefork
- worker

### PidFile

<table>
<thead>
<tr>
<th>ID</th>
<th>PidFile</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/var/run/apache.pid</td>
</tr>
</tbody>
</table>

### ErrorLog TransferLog

### PidFile

<table>
<thead>
<tr>
<th>ID</th>
<th>PidFile</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Apache 2

<table>
<thead>
<tr>
<th>ID</th>
<th>apachectl ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>ReceiveBufferSize</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>ReceiveBufferSize</td>
</tr>
</tbody>
</table>

**MPM**
- beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker

TCP

0OS
<table>
<thead>
<tr>
<th>ScoreBoardFile file-path</th>
</tr>
</thead>
<tbody>
<tr>
<td>logs/apache_status</td>
</tr>
</tbody>
</table>

- MPM
  - beos, mpm_winnt, prefork, worker

Apache

- Apache

ScoreBoardFile /var/run/apache_status

ScoreBoardFile RAM

- Apache
TCP

SendBufferSize bytes
SendBufferSize 0

MPM
beos, mpm_netware, mpm_winnt, mpmt_os2, prefork, worker

TCP

0OS
ServerLimit number

MPM

prefork, worker

prefork MPM

Apache

worker MPM

ThreadLimit

MaxClients

Apache

prefork MPM

MaxClients 256 ()

MaxClients

worker MPM

MaxClients

ThreadsPerChild 16

MaxClients

ServerLimit 20000 (prefork

Apache
**StartServers**

MPM

*mpmt_os2, prefork, worker*

```
StartServers number

MPM

mpmt_os2 2
```

StartServers 3
StartThreads number

MPM

beos, mpm_netware

mpm_netware StartThreads 50

beos StartThreads 10
<table>
<thead>
<tr>
<th>MPM</th>
<th>Apache</th>
<th>ThreadsPerChild</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpm_winnt, worker</td>
<td>Apache 2.0.41 mpm_winnt</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ThreadLimit</th>
<th>mpm_winnt</th>
<th>1920</th>
<th>64</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ThreadLimit</th>
<th>mpm_winnt</th>
<th>20000</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ThreadLimit</th>
<th>mpm_winnt</th>
<th>15000</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPM</td>
<td>mpm_winnt, worker</td>
<td>ThreadsPerChild</td>
</tr>
</tbody>
</table>
ThreadStackSize size
NetWare 65536 OS

MPM
mpm_netware, mpm_winnt, worker
2.1
User ID
User unix-userid
User #-1

MPM
prefork, worker
Apache 2.0

#
## Apache MPM beos

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>This Multi-Processing Module is optimized for BeOS.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong></td>
<td>MPM</td>
</tr>
<tr>
<td><strong>Module Identifier:</strong></td>
<td>mpm_beos_module</td>
</tr>
<tr>
<td><strong>Source File:</strong></td>
<td>beos.c</td>
</tr>
</tbody>
</table>

### Summary

This Multi-Processing Module (MPM) is the default for BeOS. It uses a single control process which creates threads to handle requests.

### See also

[Setting which addresses and ports Apache uses](#)
### Description:
Limit on the number of requests that an individual thread will handle during its life

### Syntax:
MaxRequestsPerThread number

### Default:
MaxRequestsPerThread 0

### Context:
server config

### Status:
MPM

### Module:
beos

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

Setting `MaxRequestsPerThread` to a non-zero limit has two beneficial effects:

- it limits the amount of memory that a thread can consume by (accidental) memory leakage;
- by giving threads a finite lifetime, it helps reduce the number of threads when the server load reduces.

### Note:
For `KeepAlive` requests, only the first request is counted towards this limit. In effect, it changes the behavior to limit the number of `connections` per thread.
Apache MPM event

| Description: | An experimental variant of the standard worker MPM |
| Status: | MPM |
| Module Identifier: | mpm_event_module |
| Source File: | event.c |

Summary

Warning

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

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

To use the event MPM, add --with-mpm=event to the configure script's arguments when building the httpd.

See also

The worker MPM
This MPM tries to fix the 'keep alive problem' in HTTP. After a client completes the first request, the client can keep the connection open, and send further requests using the same socket. This can save significant overhead in creating TCP connections. However, Apache traditionally keeps an entire child process/thread waiting for data from the client, which brings its own disadvantages. To solve this problem, this MPM uses a dedicated thread to handle both the Listening sockets, and all sockets that are in a Keep Alive state.

The MPM assumes that the underlying apr_pollset implementation is reasonably threadsafe. This enables the MPM to avoid excessive high level locking, or having to wake up the listener thread in order to send it a keep-alive socket. This is currently only compatible with KQueue and EPoll.
This MPM depends on **APR**'s atomic compare-and-swap operations for thread synchronization. If you are compiling for an x86 target and you don't need to support 386s, or you are compiling for a SPARC and you don't need to run on pre-UltraSPARC chips, add `--enable-nonportable-atomics=yes` to the `configure` script's arguments. This will cause APR to implement atomic operations using efficient opcodes not available in older CPUs.

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

- To use this MPM on FreeBSD, FreeBSD 5.3 or higher is recommended. However, it is possible to run this MPM on FreeBSD 5.2.1, if you use libkse (see man `libmap.conf`).
- For NetBSD, at least version 2.0 is recommended.
- For Linux, a 2.6 kernel is recommended. It is also necessary to ensure that your version of glibc has been compiled with support for EPoll.
Apache MPM netware

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

Summary

This Multi-Processing Module (MPM) implements an exclusively threaded web server that has been optimized for Novell NetWare.

The main thread is responsible for launching child worker threads which listen for connections and serve them when they arrive. Apache always tries to maintain several spare or idle worker threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new child threads to be spawned before their requests can be served.

The StartThreads, MinSpareThreads, MaxSpareThreads, and MaxThreads regulate how the main thread creates worker threads to serve requests. In general, Apache is very self-regulating, so most sites do not need to adjust these directives from their default values. Sites with limited memory may need to decrease MaxThreads to keep the server from thrashing (spawning and terminating idle threads). More information about tuning process creation is provided in the performance hints documentation.

MaxRequestsPerChild controls how frequently the server recycles processes by killing old ones and launching new ones. On the NetWare OS it is highly recommended that this directive remain set to 0. This allows worker threads to continue servicing requests.
indefinitely.

**See also**

[Setting which addresses and ports Apache uses](#)
**Description:** Set the maximum number of worker threads

**Syntax:**
```
MaxThreads number
``` 

**Default:**
MaxThreads 2048

**Context:**
server config

**Status:**
MPM

**Module:**
mpm_netware

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

```
MaxThreads 512
```
Apache MPM os2

<table>
<thead>
<tr>
<th>Description:</th>
<th>Hybrid multi-process, multi-threaded MPM for OS/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>MPM</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>mpm_mpmt_os2_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mpmt_os2.c</td>
</tr>
</tbody>
</table>

Summary

The Server consists of a main, parent process and a small, static number of child processes.

The parent process's job is to manage the child processes. This involves spawning children as required to ensure there are always `StartServers` processes accepting connections.

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

See also

- [Setting which addresses and ports Apache uses](#)
Apache HTTP 2.2
Apache MPM prefork

- fork
- MPM
- mpm_prefork_module
- prefork.c

(MPM) Unix Apache 1.3

MPM

MPM MPM

Apache
listen

MaxClients Apache

Unix 80 root
Apache

MaxRequestsPerChild
MaxSpareServers *number*

MaxSpareServers 10

MPM

prefork

- MinSpareServers
- StartServers
- **MinSpareServers** number
  - MinSpareServers 5

- MPM
- prefork

**MaxSpareServers**

1 1

- **MaxSpareServers**
- **StartServers**
Apache MPM winnt

Windows NT

MPM

mpm_winnt_module

mpm_winnt.c

(MPM)

Windows NT
Win32DisableAcceptEx:

AcceptEx():

MPM:

mpm_winnt:

2.0.49

AcceptEx() Microsoft WinSock v2 API

BSD a

AcceptEx() Windows VPN

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

- MPM
- mpm_worker_module
- worker.c

(MPM)

ThreadsPerChild
MaxClients
Apache

ThreadsPerChild

ThreadsPerChild

- MaxRequestsPerChild 0
- MaxSpareThreads

worker MPM

Unix 80

Apache

MaxRequestsPerChild
Apache mod_actions

:: CGI
:: Base
:: actions_module
:: mod_actions.c

Action
MIME
CGI

mod_cgi
CGI
Apache
# Requests for files of a particular MIME content type:
Action image/gif /cgi-bin/images.cgi

# Files of a particular file extension
AddHandler my-file-type .xyz
Action my-file-type /cgi-bin/program.cgi

virtual
• AddHandler
Script method cgi-script

AddHandler CGI URL-path
PATH_INFO PATH_TRANSLATED

# For <ISINDEX>-style searching
Script GET /cgi-bin/search

# A CGI PUT handler
Script PUT /~bob/put.cgi
Apache mod_alias

::
:: Base
:: alias_module
:: mod_alias.c

ScriptAlias CGI

Redirect

mod_alias

mod_rewrite
<VirtualHost>
  Alias /foo/bar 
  Alias /foo 
  RedirectMatch
  Alias /foo/bar 
  Alias /foo 

  /foo Alias /foo/bar Alias
</VirtualHost>
| Alias | URL-path file-path|directory-path |
|-------|------------------|
|       | URL              |
|       | Alias URL-path file-path|directory-path |
|       | ,                |
|       | Base             |
|       | mod_alias        |

**Alias**

directory-filename

http://myserver/image/foo.gif
http://myserver/imagefoo.gif

| url-path | / | /usr/local/apache/icons/ |

```xml
<Directory>
  (  <Location>

  Alias DocumentRoot

  Alias /image /ftp/pub/image
</Directory>
```

```xml
<Directory /ftp/pub/image>
  Order allow,deny
  Allow from all
</Directory>
```
**URI**

**AliasMatch** `regex file-path|directory-path`

**Base**

**mod_alias**

### Alias

**AliasMatch** `^/icons(.*) /usr/local/apache/icons$1`
Redirect URL

URL-path (%)

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

Redirect Alias ScriptAlias

status "temporary" (HTTP 302)
HTTP:
permanent (301)

temp (302)
seeother
  "See Other" (303)

gone
  "Gone" (410)

Status 300 399
(http_protocol.c send_error_response)

Redirect permanent /one http://example.com/two
Redirect 303 /three http://example.com/other
RedirectMatch 

JPEG:

RedirectMatch (.*).gif$ http://www.anotherserver.com$1.jpg
<table>
<thead>
<tr>
<th>URL</th>
<th>RedirectPermanent URL-path URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>. . . . . . .</td>
<td>FileInfo</td>
</tr>
<tr>
<td>. . . . . . .</td>
<td>Base</td>
</tr>
<tr>
<td>. . . . . . .</td>
<td>mod_alias</td>
</tr>
</tbody>
</table>

Redirect (301)
RedirectTemp URL-path URL, .htaccess, FileInfo, Base, mod_alias

Redirect (302)
<table>
<thead>
<tr>
<th>URL</th>
<th>CGI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScriptAlias</td>
<td>URL-path file-path</td>
</tr>
<tr>
<td>,</td>
<td>Base</td>
</tr>
<tr>
<td>_mod_alias</td>
<td></td>
</tr>
</tbody>
</table>

### ScriptAlias

**mod_cgi**

{scriptalias} mod_cgi
cgi-script

**URL (%)**

{scriptalias} /cgi-bin/ /web/cgi-bin/

http://myserver/cgi-bin/foo
### URL CGI

<table>
<thead>
<tr>
<th>ScriptAliasMatch</th>
<th>regex file-path</th>
<th>directory-path</th>
</tr>
</thead>
</table>

,  

Base  

| ScriptAlias | mod_alias |

bin:  

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

Copyright 2017 The Apache Software Foundation.

Apache License, Version 2.0
Apache HTTP 2.2
Apache mod_asis

send-as-is HTTP

Cgi nph

mime httpd/send-as-is

mod_headers
mod_cern_meta
Apache
send-as-is

AddHandler send-as-is asis

.asis Apache

CGI Status: HTTP

as is ()

Status: 301 Now where did I leave that URL
Location: http://xyz.abc.com/foo/bar.html
Content-type: text/html

<html>
<head>
<title>Lame excuses'R'us</title>
</head>
<body>
<h1>Fred's exceptionally wonderful page has moved to <a href="http://xyz.abc.com/foo/bar.html">Joe's</a> site.</h1>
</body>
</html>
Apache mod_auth_basic

Base
auth_basic_module
mod_auth_basic.c
Apache 2.1

AuthName
AuthType

HTTP
AuthBasicAuthoritative Off
(modules.c)

ID Required

NCSA
AuthBasicProvider

mod_authn_file

Example

```html
<Location /secure>
  AuthBasicProvider dbm
  AuthDBMType SDBM
  AuthDBMUserFile /www/etc/dbmpasswd
  Require valid-user
</Location>
```

mod_authn_dbm  mod_authn_file

Off
Apache Module mod_auth_digest

**Description:** User authentication using MD5 Digest Authentication.

**Status:** Extension

**Module Identifier:** auth_digest_module

**Source File:** mod_auth_digest.c

**Summary**
This module implements HTTP Digest Authentication ([RFC2617](https://tools.ietf.org/html/rfc2617)), and provides an alternative to [mod_auth_basic](https://httpd.apache.org/docs/current/mod/mod_auth_basic.html) where the password is not transmitted as cleartext. However, this does **not** lead to a significant security advantage over basic authentication. On the other hand, the password storage on the server is much less secure with digest authentication than with basic authentication. Therefore, using basic auth and encrypting the whole connection using [mod_ssl](https://httpd.apache.org/docs/current/mod/mod_ssl.html) is a much better alternative.

**See also**
- AuthName
- AuthType
- Require
- Satisfy
- Authentication howto
Using MD5 Digest authentication is very simple. Simply set up authentication normally, using AuthType Digest and AuthDigestProvider instead of the normal AuthType Basic and AuthBasicProvider. Then add a AuthDigestDomain directive containing at least the root URI(s) for this protection space.

Appropriate user (text) files can be created using the htdigest tool.

Example:

```xml
<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>
```

Note

Digest authentication was intended to be more secure than basic authentication, but no longer fulfills that design goal. A man-in-the-middle attacker can trivially force the browser to downgrade to basic authentication. And even a passive eavesdropper can brute-force the password using today's graphics hardware, because the hashing algorithm used by digest authentication is too fast. Another problem is that the storage of the passwords on the server is insecure. The contents of a stolen htdigest file can be used directly for digest authentication. Therefore using mod_ssl to encrypt the whole connection is strongly recommended.
**Description:** Selects the algorithm used to calculate the challenge and response hashes in digest authentication

**Syntax:** AuthDigestAlgorithm MD5|MD5-sess

**Default:** AuthDigestAlgorithm MD5

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_auth_digest

The AuthDigestAlgorithm directive selects the algorithm used to calculate the challenge and response hashes.

MD5-sess is not correctly implemented yet.
**Description:** URIs that are in the same protection space for digest authentication

**Syntax:**
```
AuthDigestDomain URI [URI] ...
```

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_auth_digest

The **AuthDigestDomain** directive allows you to specify one or more URIs which are in the same protection space (*i.e.* use the same realm and username/password info). The specified URIs are prefixes; the client will assume that all URIs "below" these are also protected by the same username/password. The URIs may be either absolute URIs (*i.e.* including a scheme, host, port, etc.) or relative URIs.

This directive *should* always be specified and contain at least the (set of) root URI(s) for this space. Omitting to do so will cause the client to send the Authorization header for every request sent to this server. Apart from increasing the size of the request, it may also have a detrimental effect on performance if **AuthDigestNcCheck** is on.

The URIs specified can also point to different servers, in which case clients (which understand this) will then share username/password info across multiple servers without prompting the user each time.
| **Description:** | Enables or disables checking of the nonce-count sent by the server |
| **Syntax:** | AuthDigestNcCheck  On|Off |
| **Default:** | AuthDigestNcCheck  Off |
| **Context:** | server config |
| **Status:** | Extension |
| **Module:** | mod_auth_digest |

Not implemented yet.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Determines how the nonce is generated</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>AuthDigestNonceFormat <em>format</em></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_auth_digest</td>
</tr>
</tbody>
</table>

Not implemented yet.
**Description:** How long the server nonce is valid

**Syntax:** AuthDigestNonceLifetime *seconds*

**Default:** AuthDigestNonceLifetime 300

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_auth_digest

The **AuthDigestNonceLifetime** directive controls how long the server nonce is valid. When the client contacts the server using an expired nonce the server will send back a 401 with `stale=true`. If `seconds` is greater than 0 then it specifies the amount of time for which the nonce is valid; this should probably never be set to less than 10 seconds. If `seconds` is less than 0 then the nonce never expires.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Sets the authentication provider(s) for this location</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>AuthDigestProvider <em>provider-name</em>&lt;br&gt;[<em>provider-name</em>] ...</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>AuthDigestProvider file</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_auth_digest</td>
</tr>
</tbody>
</table>

The **AuthDigestProvider** directive sets which provider is used to authenticate the users for this location. The default **file** provider is implemented by the `mod_authn_file` module. Make sure that the chosen provider module is present in the server.

See `mod_authn_dbm`, `mod_authn_file`, and `mod_authn_dbd` for providers.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Determines the quality-of-protection to use in digest authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>AuthDigestQop none</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>AuthDigestQop auth</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_auth_digest</td>
</tr>
</tbody>
</table>

The `AuthDigestQop` directive determines the *quality-of-protection* to use. *auth* will only do authentication (username/password); *auth-int* is authentication plus integrity checking (an MD5 hash of the entity is also computed and checked); *none* will cause the module to use the old RFC-2069 digest algorithm (which does not include integrity checking). Both *auth* and *auth-int* may be specified, in which case the browser will choose which of these to use. *none* should only be used if the browser for some reason does not like the challenge it receives otherwise.

**auth-int** is not implemented yet.
The **AuthDigestShmemSize** directive defines the amount of shared memory, that will be allocated at the server startup for keeping track of clients. Note that the shared memory segment cannot be set less than the space that is necessary for tracking at least one client. This value is dependant on your system. If you want to find out the exact value, you may simply set **AuthDigestShmemSize** to the value of 0 and read the error message after trying to start the server.

The *size* is normally expressed in Bytes, but you may let the number follow a K or an M to express your value as KBytes or MBytes. For example, the following directives are all equivalent:

```
AuthDigestShmemSize 1048576
AuthDigestShmemSize 1024K
AuthDigestShmemSize 1M
```
Apache Module mod_authn_alias

| **Description:** | Provides the ability to create extended authentication providers based on actual providers |
| **Status:** | Extension |
| **Module Identifier:** | authn_alias_module |
| **Source File:** | mod_authn_alias.c |
| **Compatibility:** | Available in Apache 2.1 and later |

**Summary**

This module allows extended authentication providers to be created within the configuration file and assigned an alias name. The alias providers can then be referenced through the directives **AuthBasicProvider** or **AuthDigestProvider** in the same way as a base authentication provider. Besides the ability to create an alias an extended provider, it also allows the same extended authentication provider to be reference by multiple locations.
This example checks for passwords in two different text files.

**Checking multiple text password files**

```xml
# Check here first
<AuthnProviderAlias file file1>
   AuthUserFile /www/conf/passwords1
</AuthnProviderAlias>

# Then check here
<AuthnProviderAlias file file2>
   AuthUserFile /www/conf/passwords2
</AuthnProviderAlias>

<Directory /var/web/pages/secure>
   AuthBasicProvider file1 file2
       AuthType Basic
       AuthName "Protected Area"
       Require valid-user
</Directory>
```

The example below creates two different ldap authentication provider aliases based on the ldap provider. This allows a single authenticated location to be serviced by multiple ldap hosts:

**Checking multiple LDAP servers**

```xml
LoadModule authn_alias_module modules/mod_authn_alias.so

<AuthnProviderAlias ldap ldap-alias1>
   AuthLDAPBindDN cn=youruser,o=ctx
   AuthLDAPBindPassword yourpassword
   AuthLDAPURL ldap://ldap.host/o=ctx
</AuthnProviderAlias>

<AuthnProviderAlias ldap ldap-other-alias>
   AuthLDAPBindDN cn=yourotheruser,o=dev
   AuthLDAPBindPassword yourotherpassword
   AuthLDAPURL ldap://other.ldap.host/o=dev?cn
</AuthnProviderAlias>

Alias /secure /webpages/secure
<Directory /webpages/secure>
   Order deny,allow
```
Allow from all

AuthBasicProvider ldap-other-alias ldap-alias1

AuthType Basic
AuthName LDAP_Protected_Place
AuthzLDAPAuthoritative off
Require valid-user

</Directory>
### Description
Enclose a group of directives that represent an extension of a base authentication provider and referenced by the specified alias.

### Syntax
```
<AuthnProviderAlias baseProvider Alias> ... </AuthnProviderAlias>
```

### Context
server config

### Status
Extension

### Module
mod_authn_alias

<AuthnProviderAlias> and </AuthnProviderAlias> are used to enclose a group of authentication directives that can be referenced by the alias name using one of the directives AuthBasicProvider or AuthDigestProvider.

This directive has no affect on authorization, even for modules that provide both authentication and authorization.
Apache mod_authn_anon

- "anonymous"
- Extension
- authn_anon_module
- mod_authn_anon.c
- Apache 2.1

mod_auth_basic anonymous-ftp ID
'anonymous'

AuthBasicProvider
htpasswd

- userID
- Anonymous_NoUserID
- Anonymous_MustGiveEmail
- Anonymous_VerifyEmail
- Anonymous_LogEmail
- userID anonymous guest www test welcome

```html
<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>
```
UserID

Anonymous user [user] ...

.htaccess

AuthConfig

Extension

mod_authn_anon

UserID

' anonymous' UserID

Anonymous anonymous "Not Registered" "I don't know"

UserID "anonymous", "AnonyMous", "Not Registered", "I Don't Know"

Apache 2.1 UserID "*"
Anonymous_LogEmail

On|Off

Anonymous_LogEmail On

.htaccess

AuthConfig

Extension

mod_authn_anon

On ( )
Anonymous_MustGiveEmail On|Off
Anonymous_MustGiveEmail On
.htaccess
AuthConfig
Extension
mod_authn_anon
0n userID ( )
Anonymous_VerifyEmail 0n|0ff
Anonymous_VerifyEmail Off
.htaccess
AuthConfig
Extension
mod_authn_anon

0n
)
Apache Module mod_authn_dbd

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>User authentication using an SQL database</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module Identifier:</strong></td>
<td>authn_dbd_module</td>
</tr>
<tr>
<td><strong>Source File:</strong></td>
<td>mod_authn_dbd.c</td>
</tr>
<tr>
<td><strong>Compatibility:</strong></td>
<td>Available in Apache 2.1 and later</td>
</tr>
</tbody>
</table>

**Summary**

This module provides authentication front-ends such as `mod_auth_digest` and `mod_auth_basic` to authenticate users by looking up users in SQL tables. Similar functionality is provided by, for example, `mod_authn_file`.

This module relies on `mod_dbd` to specify the backend database driver and connection parameters, and manage the database connections.

When using `mod_auth_basic` or `mod_auth_digest`, this module is invoked via the `AuthBasicProvider` or `AuthDigestProvider` with the `dbd` value.

**See also**

- AuthName
- AuthType
- AuthBasicProvider
- AuthDigestProvider
- DBDriver
- DBDParams
This simple example shows use of this module in the context of the Authentication and DBD frameworks. Please note that you need to load an authorization module, such as `mod_authz_user`, to get it working.

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

DBDMin 4
DBDKeep 8
DBDMax 20
DBDExptime 300

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

# core authorization configuration
Require valid-user

# mod_authn_dbd SQL query to authenticate a user
AuthDBDUserPWQuery "SELECT password FROM authn WHERE user = %s"
</Directory>
```
If httpd was built against APR version 1.3.0 or higher, then whenever a query is made to the database server, all column values in the first row returned by the query are placed in the environment, using environment variables with the prefix "AUTHENTICATE_".

If a database query for example returned the username, full name and telephone number of a user, a CGI program will have access to this information without the need to make a second independent database query to gather this additional information.

This has the potential to dramatically simplify the coding and configuration required in some web applications.
**Description:** SQL query to look up a password for a user

**Syntax:**

```
AuthDBDUserPWQuery query
```

**Context:** directory

**Status:** Extension

**Module:** mod_authn_dbd

The **AuthDBDUserPWQuery** specifies an SQL query to look up a password for a specified user. The user's ID will be passed as a single string parameter when the SQL query is executed. It may be referenced within the query statement using a `%s` format specifier.

**Example**

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

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

If httpd was built against **APR** version 1.3.0 or higher, any additional column values in the first row returned by the query statement will be stored as environment variables with names of the form `AUTHENTICATE_COLUMN`.

The encrypted password format depends on which authentication frontend (e.g. **mod_auth_basic** or **mod_auth_digest** ) is being used. See **Password Formats** for more information.
The **AuthDBDUserRealmQuery** specifies an SQL query to look up a password for a specified user and realm in a digest authentication process. The user's ID and the realm, in that order, will be passed as string parameters when the SQL query is executed. They may be referenced within the query statement using %s format specifiers.

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

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

If httpd was built against **APR** version 1.3.0 or higher, any additional column values in the first row returned by the query statement will be stored as environment variables with names of the form AUTHENTICATE_COLUMN.

The encrypted password format depends on which authentication frontend (e.g. mod_auth_basic or mod_auth_digest) is being used. See [Password Formats](#) for more information.
### Apache mod_authn_dbm

- **DBM**
- **Extension**
- **authn_dbm_module**
- **mod_authn_dbm.c**
- **Apache 2.1**

- mod_auth_digest  mod_auth_basic
- mod_authn_file

- mod_auth_basic  mod_auth_digest
- AuthDigestProvider  dbm

- AuthName
- AuthType
- AuthBasicProvider
- AuthDigestProvider
AuthDBMType default|SDBM|GDBM|NDBM|DB
AuthDBMType default
.htaccess
AuthConfig
Extension
mod_authn_dbm
AuthDBMUserFile

- AuthDBMUserFile file-path
- .htaccess
- AuthConfig
- Extension
- mod_authn_dbm

Apache module dbmopen NULL DBM
Netscape NULL
Apache dbmmanage perl
Apache mod_authn_default

mod_authn_default_module
mod_authn_default.c
Apache 2.1

mod_auth_basic
AuthDefaultAuthoritative On|Off
AuthDefaultAuthoritative On
.htaccess
AuthConfig
Base
mod_authn_default

AuthDefaultAuthoritative Off

mod_authn_default
AuthDefaultAuthoritative On
Apache mod_authn_file

mod_auth_digest  mod_auth_basic  mod_authn_dbm

mod_auth_basic  mod_auth_digest
AuthBasicProvider  AuthDigestProvider  file

AuthBasicProvider
AuthDigestProvider
htpasswd
htdigest
AuthUserFile

<table>
<thead>
<tr>
<th>ID</th>
<th>src/support</th>
<th>htdj</th>
</tr>
</thead>
</table>

**AuthUserFile**

**ID** username  Filename

```
htpasswd -c Filename username
```

**Filename**  username2:

```
htpasswd Filename username2
```

():

```
HTTP  htpasswd  htdj
```

**AuthUserFile**
Apache Module mod_authnz_ldap

**Description:** Allows an LDAP directory to be used to store the database for HTTP Basic authentication.

**Status:** Extension

**Module Identifier:** authnz_ldap_module

**Source File:** mod_authnz_ldap.c

**Compatibility:** Available in version 2.1 and later

Summary

This module provides authentication front-ends such as `mod_auth_basic` to authenticate users through an LDAP directory.

`mod_authnz_ldap` supports the following features:

- Known to support the [OpenLDAP SDK](https://www.openldap.org/) (both 1.x and 2.x), [Novell LDAP SDK](https://www.novell.com/products/ldap.html) and the [iPlanet (Netscape)](https://www.oracle.com/technetwork/developer-tools/netscape/iplanet-sdk-138961.html) SDK.
- Complex authorization policies can be implemented by representing the policy with LDAP filters.
- Uses extensive caching of LDAP operations via `mod_ldap`.
- Support for LDAP over SSL (requires the Netscape SDK) or TLS (requires the OpenLDAP 2.x SDK or Novell LDAP SDK).

When using `mod_auth_basic`, this module is invoked via the `AuthBasicProvider` directive with the `ldap` value.

See also

- `mod_ldap`
- `mod_auth_basic`
- `mod_authz_user`
- `mod_authz_groupfile`
• Operation
  ▪ The Authentication Phase
  ▪ The Authorization Phase

• The Require Directives
  ▪ Require ldap-user
  ▪ Require ldap-group
  ▪ Require ldap-dn
  ▪ Require ldap-attribute
  ▪ Require ldap-filter

• Examples
  ▪ Using TLS
  ▪ Using SSL
  ▪ Exposing Login Information
  ▪ Using Microsoft FrontPage with mod_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-dn` and `ldap-group` values.

**The Authentication Phase**

During the authentication phase, `mod_authnz_ldap` searches for an entry in the directory that matches the username that the HTTP client passes. If a single unique match is found, then `mod_authnz_ldap` attempts to bind to the directory server using the DN of the entry plus the password provided by the HTTP client. Because it does a search, then a bind, it is often referred to as the search/bind phase. Here are the steps taken during the search/bind phase.

1. Generate a search filter by combining the attribute and filter provided in the `AuthLDAPURL` directive with the username passed by the HTTP client.
2. Search the directory using the generated filter. If the search does not return exactly one entry, deny or decline access.
3. Fetch the distinguished name of the entry retrieved from the search and attempt to bind to the LDAP server using the DN
and the password passed by the HTTP client. If the bind is unsuccessful, deny or decline access.

The following directives are used during the search/bind phase

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthLDAPURL</td>
<td>Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to use.</td>
</tr>
<tr>
<td>AuthLDAPBindDN</td>
<td>An optional DN to bind with during the search phase.</td>
</tr>
<tr>
<td>AuthLDAPBindPassword</td>
<td>An optional password to bind with during the search phase.</td>
</tr>
</tbody>
</table>

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 `Requireldap-filter` directive, and the search filter successfully finds a single user object that matches the dn of the authenticated user.
- otherwise, deny or decline access

Other `Require` values may also be used which may require loading additional authorization modules. Note that if you use a `Require` value from another authorization module, you will need to ensure that `AuthzLDAPAuthoritative` is set to off to allow the authorization phase to fall back to the module providing the alternate `Require` value. When no LDAP-specific `Require` directives are used, authorization is allowed to fall back to other modules as if `AuthzLDAPAuthoritative` was set to off.

- Grant access to all successfully authenticated users if there is a `Requirevalid-user` directive. (requires `mod_authz_user`)
- Grant access if there is a `Requiregroup` 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:

<table>
<thead>
<tr>
<th><strong>AuthLDAPURL</strong></th>
<th>The attribute specified in the URL is used in compare operations for the Require ldap-user operation.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AuthLDAPCompareDNOnServer</strong></td>
<td>Determines the behavior of the Require ldap-dn directive.</td>
</tr>
<tr>
<td>AuthLDAPGroupAttribute</td>
<td>Determines the attribute to use for comparisons in the Require ldap-group directive.</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AuthLDAPGroupAttributeIsDN</td>
<td>Specifies whether to use the user DN or the username when doing comparisons for the Require ldap-group directive.</td>
</tr>
</tbody>
</table>
Apache's **Require** directives are used during the authorization phase to ensure that a user is allowed to access a resource. **mod_authnz_ldap** extends the authorization types with `ldap-user, ldap-dn, ldap-group, ldap-attribute` and `ldap-filter`. Other authorization types may also be used but may require that additional authorization modules be loaded.

**Require ldap-user**

The **Require ldap-user** directive specifies what usernames can access the resource. Once **mod_authnz_ldap** has retrieved a unique DN from the directory, it does an LDAP compare operation using the username specified in the **Require ldap-user** to see if that username is part of the just-fetched LDAP entry. Multiple users can be granted access by putting multiple usernames on the line, separated with spaces. If a username has a space in it, then it must be surrounded with double quotes. Multiple users can also be granted access by using multiple **Require ldap-user** directives, with one user per line. For example, with a **AuthLDAPURL** of `ldap://ldap/o=Airius?cn` (i.e., `cn` is used for searches), the following **Require** directives could be used to restrict access:

```
Require ldap-user "Barbara Jenson"
Require ldap-user "Fred User"
Require ldap-user "Joe Manager"
```

Because of the way that **mod_authnz_ldap** handles this directive, Barbara Jenson could sign on as **Barbara Jenson, Babs Jenson** or any other `cn` that she has in her LDAP entry. Only the single **Require ldap-user** line is needed to support all values of the attribute in the user's entry.
If the uid attribute was used instead of the cn attribute in the URL above, the above three lines could be condensed to

```
Require ldap-user 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 [AuthLDAPGroupAttribute](#) and [AuthLDAPGroupAttributeIsDN](#) directives.

### Require ldap-dn

The Require ldap-dn directive allows the administrator to grant access based on distinguished names. It specifies a DN that must match for access to be granted. If the distinguished name that was retrieved from the directory server matches the distinguished name in the Require ldap-dn, then authorization is granted. Note: do not surround the distinguished name with quotes.
The following directive would grant access to a specific DN:

```
Require ldap-dn cn=Barbara Jenson, o=Airius
```

Behavior of this directive is modified by the `AuthLDAPCompareDNOnServer` directive.

**Require ldap-attribute**

The `Require ldap-attribute` directive allows the administrator to grant access based on attributes of the authenticated user in the LDAP directory. If the attribute in the directory matches the value given in the configuration, access is granted.

The following directive would grant access to anyone with the attribute `employeeType = active`

```
Require ldap-attribute employeeType=active
```

Multiple attribute/value pairs can be specified on the same line separated by spaces or they can be specified in multiple `Require ldap-attribute` directives. The effect of listing multiple attribute/values pairs is an OR operation. Access will be granted if any of the listed attribute values match the value of the corresponding attribute in the user object. If the value of the attribute contains a space, only the value must be within double quotes.

The following directive would grant access to anyone with the city attribute equal to "San Jose" or status equal to "Active"

```
Require ldap-attribute city="San Jose" status=active
```
**Require ldap-filter**

The `Require ldap-filter` directive allows the administrator to grant access based on a complex LDAP search filter. If the dn returned by the filter search matches the authenticated user dn, access is granted.

The following directive would grant access to anyone having a cell phone and is in the marketing department

```plaintext
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.

```plaintext
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.

```plaintext
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.

```plaintext
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.

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

• The next example assumes that everyone at Airius who carries an alphanumeric pager will have an LDAP attribute of
qpagePagerID. The example will grant access only to people (authenticated via their UID) who have alphanumeric pagers:

AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid??(qpagePagerID=*)
Require valid-user

- The next example demonstrates the power of using filters to accomplish complicated administrative requirements. Without filters, it would have been necessary to create a new LDAP group and ensure that the group's members remain synchronized with the pager users. This becomes trivial with filters. The goal is to grant access to anyone who has a pager, plus grant access to Joe Manager, who doesn't have a pager, but does need to access the same resource:

AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid??(|(qpagePagerID=*)(uid=jmanager))
Require valid-user

This last may look confusing at first, so it helps to evaluate what the search filter will look like based on who connects, as shown below. If Fred User connects as fuser, the filter would look like

(&(|(qpagePagerID=*)(uid=jmanager))(uid=fuser))

The above search will only succeed if fuser has a pager. When Joe Manager connects as jmanager, the filter looks like

(&(|(qpagePagerID=*)(uid=jmanager))(uid=jmanager))

The above search will succeed whether jmanager has a pager or not.
To use TLS, see the `mod_ldap` directives `LDAPTrustedClientCert`, `LDAPTrustedGlobalCert` and `LDAPTrustedMode`.

An optional second parameter can be added to the `AuthLDAPURL` to override the default connection type set by `LDAPTrustedMode`. This will allow the connection established by an `ldap://` URL to be upgraded to a secure connection on the same port.
To use SSL, see the `mod_ldap` directives `LDAPTrustedClientCert`, `LDAPTrustedGlobalCert` and `LDAPTrustedMode`.

To specify a secure LDAP server, use `ldaps://` in the `AuthLDAPURL` directive, instead of `ldap://`. 
When this module performs authentication, LDAP attributes specified in the `AuthLDAPUrl` directive are placed in environment variables with the prefix "AUTHENTICATE_".

If the attribute field contains the username, common name and telephone number of a user, a CGI program will have access to this information without the need to make a second independent LDAP query to gather this additional information.

This has the potential to dramatically simplify the coding and configuration required in some web applications.
Normally, FrontPage uses FrontPage-web-specific user/group files (i.e., the mod_authn_file and mod_authz_groupfile modules) to handle all authentication. Unfortunately, it is not possible to just change to LDAP authentication by adding the proper directives, because it will break the Permissions forms in the FrontPage client, which attempt to modify the standard text-based authorization files.

Once a FrontPage web has been created, adding LDAP authentication to it is a matter of adding the following directives to every .htaccess file that gets created in the web:

```
AuthLDAPURL "the url"
AuthGroupFile mygroupfile
Require group mygroupfile
```

**How It Works**

FrontPage restricts access to a web by adding the Require valid-user directive to the .htaccess files. The Require valid-user directive will succeed for any user who is valid as far as LDAP is concerned. This means that anybody who has an entry in the LDAP directory is considered a valid user, whereas FrontPage considers only those people in the local user file to be valid. By substituting the ldap-group with group file authorization, Apache is allowed to consult the local user file (which is managed by FrontPage) - instead of LDAP - when handling authorizing the user.

Once directives have been added as specified above, FrontPage users will be able to perform all management operations from the FrontPage client.

**Caveats**
• When choosing the LDAP URL, the attribute to use for authentication should be something that will also be valid for putting into a `mod_authn_file` user file. The user ID is ideal for this.

• When adding users via FrontPage, FrontPage administrators should choose usernames that already exist in the LDAP directory (for obvious reasons). Also, the password that the administrator enters into the form is ignored, since Apache will actually be authenticating against the password in the LDAP database, and not against the password in the local user file. This could cause confusion for web administrators.

• Apache must be compiled with `mod_auth_basic`, `mod_authn_file` and `mod_authz_groupfile` in order to use FrontPage support. This is because Apache will still use the `mod_authz_groupfile` group file for determine the extent of a user's access to the FrontPage web.

• The directives must be put in the `.htaccess` files. Attempting to put them inside `<Location>` or `<Directory>` directives won't work. This is because `mod_authnz_ldap` has to be able to grab the `AuthGroupFile` directive that is found in FrontPage `.htaccess` files so that it knows where to look for the valid user list. If the `mod_authnz_ldap` directives aren't in the same `.htaccess` file as the FrontPage directives, then the hack won't work, because `mod_authnz_ldap` will never get a chance to process the `.htaccess` file, and won't be able to find the FrontPage-managed user file.
**Description:** Determines if other authentication providers are used when a user can be mapped to a DN but the server cannot successfully bind with the user's credentials.

**Syntax:**

```
AuthLDAPBindAuthoritative off|on
```

**Default:**

```
AuthLDAPBindAuthoritative on
```

**Context:**

directory, .htaccess

**Override:** AuthConfig

**Status:**

Extension

**Module:**

mod_authnz_ldap

**Compatibility:**

Available in versions later than 2.2.14

By default, subsequent authentication providers are only queried if a user cannot be mapped to a DN, but not if the user can be mapped to a DN and their password cannot be verified with an LDAP bind. If `AuthLDAPBindAuthoritative` is set to off, other configured authentication modules will have a chance to validate the user if the LDAP bind (with the current user's credentials) fails for any reason.

This allows users present in both LDAP and `AuthUserFile` to authenticate when the LDAP server is available but the user's account is locked or password is otherwise unusable.

**See also**

- [AuthUserFile](#)
- [AuthBasicProvider](#)
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Optional DN to use in binding to the LDAP server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>AuthLDAPBindDN  <code>distinguished-name</code></td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>mod_authnz_ldap</td>
</tr>
</tbody>
</table>

An optional DN used to bind to the server when searching for entries. If not provided, `mod_authnz_ldap` will use an anonymous bind.
**Description:** Password used in conjunction with the bind DN

**Syntax:**

`AuthLDAPBindPassword` *password*

**Context:**

directory, .htaccess

**Override:**

AuthConfig

**Status:**

Extension

**Module:**

`mod_authnz_ldap`

**Compatibility:**

exec: was added in 2.2.25.

A bind password to use in conjunction with the bind DN. Note that the bind password is probably sensitive data, and should be properly protected. You should only use the `AuthLDAPBindDN` and `AuthLDAPBindPassword` if you absolutely need them to search the directory.

If the value begins with exec: the resulting command will be executed and the first line returned to standard output by the program will be used as the password.

```plaintext
#Password used as-is
AuthLDAPBindPassword secret

#Run /path/to/program to get my password
AuthLDAPBindPassword exec:/path/to/program

#Run /path/to/otherProgram and provide arguments
AuthLDAPBindPassword "exec:/path/to/otherProgram argument1"
```
The `AuthLDAPCharsetConfig` directive sets the location of the language to charset conversion configuration file. `File-path` is relative to the `ServerRoot`. This file specifies the list of language extensions to character sets. Most administrators use the provided `charset.conv` file, which associates common language extensions to character sets.

The file contains lines in the following format:

```
Language-Extension charset [Language-String] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (#) are ignored.
**Description:** Use the LDAP server to compare the DNs

**Syntax:**
`AuthLDAPCompareDNOnServer on|off`

**Default:**
`AuthLDAPCompareDNOnServer on`

**Context:**
directory, .htaccess

**Override:**
AuthConfig

**Status:**
Extension

**Module:**
`mod_authnz_ldap`

When set, `mod_authnz_ldap` will use the LDAP server to compare the DNs. This is the only foolproof way to compare DNs. `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.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>When will the module de-reference aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>AuthLDAPDereferenceAliases never</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>AuthLDAPDereferenceAliases Always</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_authnz_ldap</td>
</tr>
</tbody>
</table>

This directive specifies when mod_authnz_ldap will de-reference aliases during LDAP operations. The default is always.
<table>
<thead>
<tr>
<th>Description</th>
<th>LDAP attributes used to check for group membership</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>AuthLDAPGroupAttribute <code>attribute</code></td>
</tr>
<tr>
<td>Default</td>
<td>AuthLDAPGroupAttribute <code>member</code> uniquemember</td>
</tr>
<tr>
<td>Context</td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td>Override</td>
<td>AuthConfig</td>
</tr>
<tr>
<td>Status</td>
<td>Extension</td>
</tr>
<tr>
<td>Module</td>
<td>mod_authnz_ldap</td>
</tr>
</tbody>
</table>

This directive specifies which LDAP attributes are used to check for group membership. Multiple attributes can be used by specifying this directive multiple times. If not specified, then `mod_authnz_ldap` uses the member and uniquemember attributes.
**Description:** Use the DN of the client username when checking for group membership.

**Syntax:** AuthLDAPGroupAttributeIsDN on|off

**Default:** AuthLDAPGroupAttributeIsDN on

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_authnz_ldap

When set on, this directive says to use the distinguished name of the client username when checking for group membership. Otherwise, the username will be used. For example, assume that the client sent the username bjenson, which corresponds to the LDAP DN cn=Babs Jenson, o=Airius. If this directive is set, mod_authnz_ldap will check if the group has cn=Babs Jenson, o=Airius as a member. If this directive is not set, then mod_authnz_ldap will check if the group has bjenson as a member.
**Description:** Use the value of the attribute returned during the user query to set the REMOTE_USER environment variable

**Syntax:** AuthLDAPRemoteUserAttribute uid

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_authnz_ldap

If this directive is set, the value of the REMOTE_USER environment variable will be set to the value of the attribute specified. Make sure that this attribute is included in the list of attributes in the AuthLDAPUrl definition, otherwise this directive will have no effect. This directive, if present, takes precedence over AuthLDAPRemoteUserIsDN. This directive is useful should you want people to log into a website using an email address, but a backend application expects the username as a userid.
**Description:** Use the DN of the client username to set the REMOTE_USER environment variable

**Syntax:** AuthLDAPRemoteUserIsDN on|off

**Default:** AuthLDAPRemoteUserIsDN off

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** 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.
**Description:** URL specifying the LDAP search parameters

**Syntax:**

```
AuthLDAPUrl url

[NONE | SSL | TLS | STARTTLS]
```

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_authnz_ldap

An RFC 2255 URL which specifies the LDAP search parameters to use. The syntax of the URL is

```
```

**ldap**

For regular ldap, use the string ldap. For secure LDAP, use ldaps instead. Secure LDAP is only available if Apache was linked to an LDAP library with SSL support.

**host:port**

The name/port of the ldap server (defaults to localhost:389 for ldap, and localhost:636 for ldaps). To specify multiple, redundant LDAP servers, just list all servers, separated by spaces. `mod_authnz_ldap` will try connecting to each server in turn, until it makes a successful connection.

Once a connection has been made to a server, that connection remains active for the life of the httpd process, or until the LDAP server goes down.

If the LDAP server goes down and breaks an existing connection, `mod_authnz_ldap` will attempt to re-connect, starting with the primary server, and trying each redundant
server in turn. Note that this is different than a true round-robin search.

**basedn**
The DN of the branch of the directory where all searches should start from. At the very least, this must be the top of your directory tree, but could also specify a subtree in the directory.

**attribute**
The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute will be used, no matter how many are provided. If no attributes are provided, the default is to use uid. It's a good idea to choose an attribute that will be unique across all entries in the subtree you will be using.

**scope**
The scope of the search. Can be either one or sub. Note that a scope of base is also supported by RFC 2255, but is not supported by this module. If the scope is not provided, or if base scope is specified, the default is to use a scope of sub.

**filter**
A valid LDAP search filter. If not provided, defaults to `(objectClass=*)`, which will search for all objects in the tree. Filters are limited to approximately 8000 characters (the definition of MAX_STRING_LEN in the Apache source code). This should be more than sufficient for any application.

When doing searches, the attribute, filter and username passed by the HTTP client are combined to create a search filter that looks like `(&(filter)(attribute=username))`.

For example, consider an URL of `ldap://ldap.airius.com/o=Airius?cn?sub?`
When a client attempts to connect using a username of Babs Jenson, the resulting search filter will be `(& (posixid=*)(cn=Babs Jenson))`.

An optional parameter can be added to allow the LDAP URL to override the connection type. This parameter can be one of the following:

**NONE**

Establish an unsecure connection on the default LDAP port. This is the same as `ldap://` on port 389.

**SSL**

Establish a secure connection on the default secure LDAP port. This is the same as `ldaps://`

**TLS | STARTTLS**

Establish an upgraded secure connection on the default LDAP port. This connection will be initiated on port 389 by default and then upgraded to a secure connection on the same port.

See above for examples of `AuthLDAPURL` URLs.

When `AuthLDAPURL` is enabled in a particular context, but some other module has performed authentication for the request, the server will try to map the username to a DN during authorization regardless of whether or not LDAP-specific requirements are present. To ignore the failures to map a username to a DN during authorization, set `AuthzLDAPAuthoritative` to "off".
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Prevent other authentication modules from authenticating the user if this one fails</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>AuthzLDAPAuthoritative on</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>AuthzLDAPAuthoritative on</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_authnz_ldap</td>
</tr>
</tbody>
</table>

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

When no LDAP-specific `Require` directives are used, authorization is allowed to fall back to other modules as if `AuthzLDAPAuthoritative` was set to `off`. 
Apache Module mod_authz_dbm

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Group authorization using DBM files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module Identifier:</strong></td>
<td>authz_dbm_module</td>
</tr>
<tr>
<td><strong>Source File:</strong></td>
<td>mod_authz_dbm.c</td>
</tr>
<tr>
<td><strong>Compatibility:</strong></td>
<td>Available in Apache 2.1 and later</td>
</tr>
</tbody>
</table>

**Summary**

This module provides authorization capabilities so that authenticated users can be allowed or denied access to portions of the web site by group membership. Similar functionality is provided by `mod_authz_groupfile`.

**See also**

- `Require`
- `Satisfy`
**Description:** Sets the name of the database file containing the list of user groups for authorization.

**Syntax:**
```
AuthDBMGroupFile file-path
```

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_authz_dbm

The **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.

**Security**

Make sure that the **AuthDBMGroupFile** is stored outside the document tree of the web-server. Do not put it in the directory that it protects. Otherwise, clients will be able to download the **AuthDBMGroupFile** unless otherwise protected.

Combining Group and Password DBM files: In some cases it is easier to manage a single database which contains both the password and group details for each user. This simplifies any support programs that need to be written: they now only have to deal with writing to and locking a single DBM file. This can be accomplished by first setting the group and password files to point to the same DBM:

```
AuthDBMGroupFile /www/userbase
```
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.
**Description:** Sets whether authorization will be passed on to lower level modules

**Syntax:** AuthzDBMAuthoritative On|Off

**Default:** AuthzDBMAuthoritative On

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** 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 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 AuthAuthoritative 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.

**Security**
Do consider the implications of allowing a user to allow fall-through in his .htaccess file; and verify that this is really what you want; Generally it is easier to just secure a single .htpasswd file, than it is to secure a database which might have more access interfaces.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Sets the type of database file that is used to store the list of user groups</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>AuthzDBMType default</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>AuthzDBMType default</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>AuthConfig</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_authz_dbm</td>
</tr>
</tbody>
</table>

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 mod_authz_default

```
:: Base
:: authz_default_module
:: mod_authz_default.c
:: Apache 2.1
```

mod_authz_user  mod_authz_groupfile
<table>
<thead>
<tr>
<th>AuthzDefaultAuthoritative</th>
<th>On</th>
<th>Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthzDefaultAuthoritative</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>.htaccess</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AuthConfig</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mod_authz_default</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AuthzDefaultAuthoritative: Off

<table>
<thead>
<tr>
<th>mod_authz_default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthDefaultAuthoritative: (On)</td>
</tr>
</tbody>
</table>
Apache mod_authz_groupfile

Base
authz_groupfile_module
mod_authz_groupfile.c
Apache 2.1

Require
Satisfy
AuthGroupFile

AuthGroupFile

mygroup: bob joe anne

AuthGroupFile
AuthzGroupFileAuthoritative On|Off
AuthzGroupFileAuthoritative On
.htaccess
AuthConfig
Base
mod_authz_groupfile

AuthzGroupFileAuthoritative Off
(module.c)

Authentication Required
.htaccess
Apache mod_authz_host

mod_authz_host
.htaccess IP
Order Allow Deny
(GET, PUT, POST)

Satisfy
Require
Allow

from all|host|env=env-variable [host|env=env-variable] ...

.htaccess

Limit

Base

mod_authz_host

Allow from apache.org
Allow from .net example.edu

Apache HostnameLookups IP

IP

IP

Allow from 10.1.2.3
Allow from 192.168.1.104 192.168.1.205

IP

IP
<table>
<thead>
<tr>
<th>IP Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Allow from</td>
</tr>
<tr>
<td>172.20</td>
<td>192.168.2</td>
</tr>
<tr>
<td>10.1.0.0/255.255.0.0</td>
<td>Allow from</td>
</tr>
<tr>
<td>a.b.c.d w.x.y.z</td>
<td>/nnn CIDR</td>
</tr>
<tr>
<td>10.1.0.0/16</td>
<td>Allow from</td>
</tr>
<tr>
<td>2001:db8::a00:20ff:fea7:ccea</td>
<td>Allow from</td>
</tr>
<tr>
<td>2001:db8::a00:20ff:fea7:ccea/10</td>
<td>Allow from</td>
</tr>
</tbody>
</table>

**Example:**

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

user-agent KnockKnock/2.0
Deny from all|host|env=env-variable
  [host|env=env-variable] ...

.htaccess
Limit
Base
mod_authz_host

IP
Order | Allow | Deny

Order ordering

Order Deny, Allow

.htaccess

Limit

Base

mod_authz_host

Order Allow Deny

Deny, Allow

Deny Allow

Allow Deny

Allow, Deny

Mutual-failure

Allow Deny

Allow Apache.org

Order Deny, Allow

Deny from all

Allow from apache.org

foo.apache.org Apache.org

Order Allow, Deny

Allow from apache.org

Deny from foo.apache.org

Order Deny, Allow

Apache.org Deny from foo.apache.org
<Directory /www>
    Order Allow,Deny
</Directory>

deny /www

Order
    Directory .htaccess Allow Den
    Directory,Location, Files

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0.
Apache mod_authz_owner

Extension
authz_owner_module
mod_authz_owner.c
Apache 2.1

mod_authz_owner

mod_authz_groupfile
() accounts
() "MultiViews"

HTTP ID (ID)

mod_auth_digest

file-owner file-group:

file-owner

file-group

mod_authz_groupfile
() accounts
() "MultiViews"

Require
Satisfy
Require file-owner

Apache

AuthDBMUserFile

/home/smith/public_html/private

smith

jones

<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

AuthDBMGroupFile

foo

project-foo

<Directory /home/*/public_html/project-foo>
  AuthType Basic
  AuthName "Project Foo Files"
  AuthBasicProvider dbm

  # combined user/group database
  AuthDBMUserFile /usr/local/apache2/etc/.htdbm-all
  AuthDBMGroupFile /usr/local/apache2/etc/.htdbm-all

  Satisfy All
  Require file-group
</Directory>
AuthzOwnerAuthoritative On|Off
AuthzOwnerAuthoritative On
.htaccess
AuthConfig
Extension
mod_authz_owner

AuthzOwnerAuthoritative Off

- file-owner
- file-group

Off file-owner file-group

Authentication Required

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0.
valid-user

Require
Satisfy
AuthzUserAuthoritative  On|Off

AuthzUserAuthoritative  On

.htaccess

AuthConfig

Base

mod_authz_user

AuthzUserAuthoritative  Off  userID

Authentication  Require
Apache mod_autoindex

<table>
<thead>
<tr>
<th>Unix</th>
<th>ls</th>
<th>Win32</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Base

autoindex_module

mod_autoindex.c

- index.html
- DirectoryInc
  - AddIconByType

Options +Indexes

  - FancyIndexing
  - IndexOptions
  - SuppressColumnSorting

"Size"
Apache 2.0.23

- C=N
- C=M
- C=S
- C=D

- O=A
- O=D

- F=0  (FancyIndex )
- F=1 FancyIndex
- F=2 HTML FancyIndex
- V=0
- V=1

- P=pattern pattern

"P (P)" IndexIgnore
mod_autoindex ()

HEADER.html

<form action="" method="get">
  Show me a <select name="F">
    <option value="0"> Plain list</option>
    <option value="1" selected="selected"> Fancy list</option>
    <option value="2"> Table list</option>
  </select>
  Sorted by <select name="C">
    <option value="N" selected="selected"> Name</option>
    <option value="M"> Date Modified</option>
    <option value="S"> Size</option>
  </select>
</form>
<select name="O">
    <option value="A" selected="selected"> Ascending</option>
    <option value="D"> Descending</option>
</select>
<select name="V">
    <option value="0" selected="selected"> in Normal order</option>
    <option value="1"> in Version order</option>
</select>
Matching <input type="text" name="P" value="*" />
<input type="submit" name="X" value="Go" />
</form>
AddAlt string file [file] ...
/Add, /htaccess
/Indexes
/Base
/mod_autoindex

AddAlt FancyIndexing

(" ")

AddAlt "PDF file" *.pdf
AddAlt Compressed *.gz *.zip *.Z
MIME
AddAltByEncoding string MIME-encoding [MIME-encoding] ...
., .htaccess
Indexes
Base
mod_autoindex

AddAltByEncoding FancyIndexing encoding x-compress string ( " ')

AddAltByEncoding gzip x-gzip
AddAltByType string MIME-type [MIME-type] ...

..,.htaccess
Indexes
Base
mod_autoindex

AddAltByType FancyIndexing
text/html string (""")

AddAltByType 'plain text' text/plain
AddDescription string file [file] ...
AddDescription 
AddDescription Indexes
AddDescription Base
AddDescription mod_autoindex

FancyIndexing file

AddDescription "The planet Mars" /web/pics/mars.gif

23
IndexOptions SuppressSize 7
IndexOptions SuppressLastModified 19

AddDescription HTML
AddIcon  icon  name  [name]  ...

Indexes

Base

mod_autoindex

FancyIndexing  name
(alttex, url)  alttext

name  ^^DIRECTORY^^  ^^BLANKICON^^ (char)

AddIcon (IMG,/icons/image.xbm).gif .jpg .xbm
AddIcon /icons/dir.xbm  ^^DIRECTORY^^
AddIcon /icons/backup.xbm  *~

AddIcon  AddIconByType
AddIconByEncoding icon MIME-encoding [MIME-encoding] ...

, .htaccess

Indexes

Base

mod_autoindex

FancyIndexing

(alttext, url) alttext

MIME-encoding

AddIconByEncoding /icons/compress.xbm x-compress
AddIconByType *icon* MIME-type [MIME-type] ...

,,.htaccess
Indexes
Base
mod_autoindex

**FancyIndexing**

*(alttext, url) alttext*

**MIME-type**

AddIconByType (IMG,/icons/image.xbm) image/*
: 
  : DefaultValue `url-path`
  : .htaccess
  : Indexes
  : Base
  : `mod_autoindex`

**FancyIndexing**

DefaultIcon `/icon/unknown.xbm`
HeaderName

HeaderName filename

HeaderName .htaccess

HeaderName Indexes

HeaderName Base

HeaderName mod_autoindex

HeaderName

HeaderName HEADER.html

HeaderName

HeaderName ReadmeName filename URI

HeaderName filename DocumentRoot

HeaderName /include/HEADER.html

HeaderName

HeaderName " text/*" (text/html, text/CGI

HeaderName AddType text/html .cgi

HeaderName Options MultiViews

HeaderName text/html options Includes Include

HeaderName (mod_include)

HeaderName

HeaderName HTML (<html>, <head>,

IndexOptions +SuppressHTMLPreamble
<table>
<thead>
<tr>
<th>Inserts text in the HEAD section of an index page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexHeadInsert &quot;markup ...&quot;</td>
</tr>
<tr>
<td>. . . . . . . .htaccess</td>
</tr>
<tr>
<td>Indexes</td>
</tr>
<tr>
<td>Base</td>
</tr>
<tr>
<td>mod_autoindex</td>
</tr>
<tr>
<td>Available in Apache 2.2.11 and later</td>
</tr>
</tbody>
</table>
IndexIgnore
()

IndexIgnore README .htaccess *.bak *~
IndexOptions

**DescriptionWidth=[n | *] (2.0.23)**

- DescriptionWidth
  - DescriptionWidth ()

- mod_autoindex
  - DescriptionWidth=n
  - DescriptionWidth=*  
  - AddDescription

**FancyIndexing**

**FoldersFirst (2.0.23)**

- Zed  
  - Beta
  - Gamma

**HTMLTable (Apache 2.0.23)**

- FancyIndexing

**IconsAreLinks**

- FancyIndexing

**IconHeight[=pixels]**

- IconWidth
  - Apache

**IconWidth[=pixels]**

- IconHeight
IgnoreCase

gamma )

IgnoreClient

    mod autoindex
    SuppressColumnSorting )

NameWidth=[n | *]

    NameWidth
    -NameWidth ()
    NameWidth=n n
    NameWidth=*

ScanHTMLTitles

    FancyIndexing HTML
    httpd title CPU (:

ShowForbidden

    HTTP_UNAUTHORIZED

SuppressColumnSorting

    Apache FancyIndexing
    2.0.23
    IgnoreClient

SuppressDescription

    FancyIndexing
    AddDescription De:

SuppressHTMLPreamble

    HeaderName HTML
    SuppressHTMLPreamble

SuppressIcon (Apache 2.0.23 )
FancyIndexing
HTML 3.2
(FancyIndexing )

SuppressLastModified
FancyIndexing

SuppressRules (Apache 2.0.23 )
( hr )
HTML 3.2
(FancyIndexing )

SuppressSize
FancyIndexing

TrackModified (Apache 2.0.23 )
HTTP ETag
OS2 JFS Win32 NTFS
HEAD

VersionSort (Apache 2.0a3 )
VersionSort:

: 
foo-1.7
foo-1.7.2
foo-1.7.12
foo-1.8.2
foo-1.8.2a
foo-1.12

0

foo-1.001
foo-1.002
foo-1.030
foo-1.04

XHTML (Apache 2.0.49 )
IndexOptions
Apache 1.3.3

- IndexOptions

```html
<Directory /foo>
  IndexOptions HTMLTable
  IndexOptions SuppressColumnsorting
</Directory>
```

IndexOptions HTMLTable SuppressColumnsorting

- (`+` `-` )

`+` `-`

IndexOptions +ScanHTMLTitles -IconsAreLinks FancyIndexing
IndexOptions +SuppressSize

IndexOptions FancyIndexing +SuppressSize

FancyIndexing
<table>
<thead>
<tr>
<th>IndexOrderDefault Ascending</th>
<th>Descending Name</th>
<th>Date</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>, , , .htaccess</td>
<td>Indexes</td>
<td>Base</td>
<td>mod_autoindex</td>
<td></td>
</tr>
</tbody>
</table>

**IndexOrderDefault**  

**FancyIndexing**  

**IndexOrderDefault**  

**Name, Date, Size Description**  

**SuppressColumnSorting**
IndexStyleSheet

CSS

IndexStyleSheet url-path

IndexStyleSheet .htaccess

IndexStyleSheet Indexes

IndexStyleSheet Base

IndexStyleSheet mod_autoindex

IndexStyleSheet "/css/style.css"
<table>
<thead>
<tr>
<th></th>
<th>ReadmeName filename</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>./.htaccess</td>
</tr>
<tr>
<td></td>
<td>Indexes</td>
</tr>
<tr>
<td></td>
<td>Base</td>
</tr>
<tr>
<td></td>
<td>mod_autoindex</td>
</tr>
</tbody>
</table>

ReadmeName DocumentRoot

ReadmeName FOOTER.html

2

ReadmeName /include/FOOTER.html

HeaderName

# Apache mod_cache

<table>
<thead>
<tr>
<th>URI</th>
<th>Extension</th>
<th>Cache Module</th>
<th>Source File</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_cache</td>
<td></td>
<td></td>
<td>mod_cache.c</td>
</tr>
<tr>
<td>mod_disk_cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mod_mem_cache</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Allow**  **Deny**

**mod_cache**

- **mod_disk_cache**

- **mod_mem_cache**

  - **mod_proxy**  **ProxyPass**

**URI**

[Caching Guide](#)
<table>
<thead>
<tr>
<th>mod disk cache</th>
<th>CacheRoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod mem cache</td>
<td>CacheDirLevels</td>
</tr>
<tr>
<td></td>
<td>CacheDirLength</td>
</tr>
<tr>
<td></td>
<td>CacheMinFileSize</td>
</tr>
<tr>
<td></td>
<td>CacheMaxFileSize</td>
</tr>
<tr>
<td></td>
<td>MCacheSize</td>
</tr>
<tr>
<td></td>
<td>MCacheMaxObjectCount</td>
</tr>
<tr>
<td></td>
<td>MCacheMinObjectSize</td>
</tr>
<tr>
<td></td>
<td>MCacheMaxObjectSize</td>
</tr>
<tr>
<td></td>
<td>MCacheRemovalAlgorithm</td>
</tr>
<tr>
<td></td>
<td>MCacheMaxStreamingBuffer</td>
</tr>
</tbody>
</table>
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>
</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 **seconds**

CacheDefaultExpire 3600 (1)

CacheDefaultExpire 86400

CacheMaxExpire
CacheDisable

**URL**

**CacheDisable**  *url-string*

,  

**Extension**

**mod_cache**

---

CacheDisable  /local_files
CacheEnable cache_type url-string

CacheEnable mod_cache url-string URL

cache_type
mod_cache
mod_cache
mod_mem_cache

() CacheEnable URL

CacheEnable mem /manual
CacheEnable fd /images
CacheEnable disk /

url-string

# 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/
CacheIgnoreCacheControl

<table>
<thead>
<tr>
<th>CacheIgnoreCacheControl</th>
<th>On</th>
<th>Off</th>
</tr>
</thead>
</table>

Pragma: no-store

Cache-Control: no-cache

- **CacheStorePrivate**
- **CacheStoreNoStore**
CacheIgnoreHeaders header-string [header-string] ...

CacheIgnoreHeaders None

Extension

mod_cache

RFC 2616 hop-by-hop HTTP

CacheIgnoreHeaders:

- Connection
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade

CacheIgnoreHeaders HTTP

CacheIgnoreHeaders HTTP

(RFC 2616) CacheIgnoreHeaders None

1

CacheIgnoreHeaders Set-Cookie

2

CacheIgnoreHeaders None
: Expires
  CacheIgnore
<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Modified</td>
<td></td>
</tr>
<tr>
<td>CacheIgnoreNoLastMod</td>
<td>On</td>
</tr>
<tr>
<td>CacheIgnoreNoLastMod</td>
<td>Off</td>
</tr>
<tr>
<td>Extension</td>
<td></td>
</tr>
<tr>
<td>mod_cache</td>
<td></td>
</tr>
</tbody>
</table>

Last-Modified (expiry)

CacheIgnoreNoLastMod

CacheDef
Cache Ignore Query String

- Ignore query string when caching
- CacheIgnoreQueryString On|Off
- CacheIgnoreQueryString Off

Extension
- mod_cache

Available in Apache 2.2.6 and later
Ignore defined session identifiers encoded in the URL when caching

CacheIgnoreURLSessionIdentifiers identifier [identifier] ...

CacheIgnoreURLSessionIdentifiers None

Extension

mod_cache
Last-Modified (expiry)

CacheLastModifiedFactor

expiry-period = time-since-last-modified-date * factor
expiry-date = current-date + expiry-period

10 * factor 0.1 = 10 * 0.1 = 1
3:00pm + 1hour = 4:00pm

CacheMaxExpire

CacheLastModifiedFactor 0.5
Enable the thundering herd lock.

CacheLock on/off

CacheLock off

Extension

mod_cache

Available in Apache 2.2.15 and later
Set the maximum possible age of a cache lock.

CacheLockMaxAge  integer

CacheLockMaxAge  5

Extension

mod_cache
Set the lock path directory.

CacheLockPath directory

CacheLockPath /tmp/mod_cache-lock

Extension

mod_cache
<table>
<thead>
<tr>
<th></th>
<th>CacheMaxExpire</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>seconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td>86400 (</td>
<td></td>
</tr>
<tr>
<td>Extension</td>
<td>mod_cache</td>
<td></td>
</tr>
<tr>
<td>CacheMaxExpire</td>
<td>604800</td>
<td></td>
</tr>
</tbody>
</table>
Cache-Control: no-store

CacheStoreNoStore On

CacheIgnoreCacheControl

CacheStorePrivate
| private |
| CacheStorePrivate On|Off |
| CacheStorePrivate Off |

Cache-Control: private

CacheStorePrivate On private

- CacheIgnoreCacheControl
- CacheStoreNoStore
Apache Module mod_cern_meta

<table>
<thead>
<tr>
<th>Description:</th>
<th>CERN httpd metafile semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>Extension</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>cern_meta_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mod_cern_meta.c</td>
</tr>
</tbody>
</table>

Summary

Emulate the CERN HTTPD Meta file semantics. Meta files are HTTP headers that can be output in addition to the normal range of headers for each file accessed. They appear rather like the Apache .asis files, and are able to provide a crude way of influencing the Expires: header, as well as providing other curiosities. There are many ways to manage meta information, this one was chosen because there is already a large number of CERN users who can exploit this module.

More information on the [CERN metafile semantics](#) is available.

See also

- [mod_headers](#)
- [mod_asis](#)
**Description:** Name of the directory to find CERN-style meta information files

**Syntax:** MetaDir *directory*

**Default:** MetaDir .web

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

**Override:** Indexes

**Status:** Extension

**Module:** mod_cern_meta

Specifies the name of the directory in which Apache can find meta information files. The directory is usually a 'hidden' subdirectory of the directory that contains the file being accessed. Set to "." to look in the same directory as the file:

```bash
MetaDir .
```

Or, to set it to a subdirectory of the directory containing the files:

```bash
MetaDir .meta
```
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Activates CERN meta-file processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>MetaFiles on</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>MetaFiles off</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host, directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>Indexes</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_cern_meta</td>
</tr>
</tbody>
</table>

Turns on/off Meta file processing on a per-directory basis.
**Description:** File name suffix for the file containing CERN-style meta information

**Syntax:** MetaSuffix suffix

**Default:** MetaSuffix .meta

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

**Override:** Indexes

**Status:** Extension

**Module:** mod_cern_meta

Specifies the file name suffix for the file containing the meta information. For example, the default values for the two directives will cause a request to DOCUMENT_ROOT/somedir/index.html to look in DOCUMENT_ROOT/somedir/.web/index.html.meta and will use its contents to generate additional MIME header information.

**Example:**

MetaSuffix .meta
Apache mod_cgi

- CGI
- Base
- cgi_module
- mod_cgi.c

cgi-script CGI
ScriptAlias

Apache CGI
Unix MPM
MIME application/x-httpd-cgi

AcceptPathInfo
Options
ScriptAlias
AddHandler
CGI ID
CGI
CGI

**PATH_INFO**

- **AcceptPathInfo**: off
- **mod_cgi**: (URI /more)
- **NOT FOUND**: AcceptPathInfo

**REMOTE_HOST**

- **HostnameLookups**: on (off)
- **DNS**

**REMOTE_IDENT**

- **IdentityCheck**: on
- **ieder**

**REMOTE_USER**

- **CGI**
%% [time] request-line
%% HTTP-status CGI-script-filename

%%error
error-message

%request
HTTP
() POST PUT
%response
CGI
%stdout
CGI
%stderr
CGI

()  %stdout %stderr
ScriptLog

<table>
<thead>
<tr>
<th>CGI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScriptLog <em>file-path</em></td>
</tr>
<tr>
<td>,</td>
</tr>
<tr>
<td>Base</td>
</tr>
<tr>
<td><em>mod_cgi, mod_cgid</em></td>
</tr>
</tbody>
</table>

User

CGI

ScriptLog logs/cgi_log
PUT POST

ScriptLogBuffer bytes

ScriptLogBuffer 1024

Base

mod_cgi, mod_cgid

PUT POST
ScriptLogLength:

<table>
<thead>
<tr>
<th>CGI</th>
<th>ScriptLogLength bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ScriptLogLength 10385760</td>
</tr>
</tbody>
</table>

Base: mod_cgi, mod_cgid

ScriptLogLength CGI

CGI
Apache mod_cgid

: CGI
  CGI
: Base
cgid_module
: mod_cgid.c
: Unix
  MPM

ScriptSock  mod_cgid  mod_cgi
CGI

Unix
fork
unix

MPM

CGI
mod_cgi
CGI_ID
The length of time to wait for more output from the CGI program.

`CGIDScriptTimeout time[s|ms]` is the value of the `Timeout` directive when unset.

In `.htaccess` files, `mod_cgid`.

`CGIDScriptTimeout` defaults to zero in releases 2.4 and earlier.
CGI

ScriptSock file-path

ScriptSock logs/cgisock

Base

mod_cgid

ScriptSock /var/run/cgid.sock
Apache Module mod_charset_lite

**Description:** Specify character set translation or recoding

**Status:** Extension

**Module Identifier:** charset_lite_module

**Source File:** mod_charset_lite.c

**Summary**

`mod_charset_lite` allows the server to change the character set of responses before sending them to the client. In an EBCDIC environment, Apache always translates HTTP protocol content (e.g., response headers) from the code page of the Apache process locale to ISO-8859-1, but not the body of responses. In any environment, `mod_charset_lite` can be used to specify that response bodies should be translated. For example, if files are stored in EBCDIC, then `mod_charset_lite` can translate them to ISO-8859-1 before sending them to the client.

This module provides a small subset of configuration mechanisms implemented by Russian Apache and its associated `mod_charset`. 
Invalid character set names

The character set name parameters of `CharsetSourceEnc` and `CharsetDefault` must be acceptable to the translation mechanism used by APR on the system where `mod_charset_lite` is deployed. These character set names are not standardized and are usually not the same as the corresponding values used in http headers. Currently, APR can only use `iconv(3)`, so you can easily test your character set names using the `iconv(1)` program, as follows:

```bash
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.
**CharsetDefault**

<table>
<thead>
<tr>
<th>Description</th>
<th>Charset to translate into</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>CharsetDefault charset</code></td>
</tr>
<tr>
<td>Context</td>
<td>server config, virtual host, directory, .htaccess</td>
</tr>
<tr>
<td>Override</td>
<td>FileInfo</td>
</tr>
<tr>
<td>Status</td>
<td>Extension</td>
</tr>
<tr>
<td>Module</td>
<td>mod_charset_lite</td>
</tr>
</tbody>
</table>

The **CharsetDefault** directive specifies the charset that content in the associated container should be translated to.

The value of the `charset` argument must be accepted as a valid character set name by the character set support in [APR](https://httpd.apache.org/docs/2.4/httpd.html#iconv). Generally, this means that it must be supported by `iconv`.

**Example**

```xml
<Directory /export/home/trawick/apacheinst/htdocs/convert>
  CharsetSourceEnc UTF-16BE
  CharsetDefault ISO-8859-1
</Directory>
```
| **Description:** | Configures charset translation behavior |
| **Syntax:** | `CharsetOptions option [option] ...` |
| **Default:** | `CharsetOptions DebugLevel=0`  
| NoImplicitAdd |
| **Context:** | server config, virtual host, directory, .htaccess |
| **Override:** | FileInfo |
| **Status:** | Extension |
| **Module:** | `mod_charset_lite` |

The `CharsetOptions` directive configures certain behaviors of `mod_charset_lite`. *Option* can be one of

### DebugLevel=\n
The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_charset_lite`. By default, no messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the DBGLVL_ constants near the beginning of `mod_charset_lite.c`.

### ImplicitAdd | NoImplicitAdd

The `ImplicitAdd` keyword specifies that `mod_charset_lite` should implicitly insert its filter when the configuration specifies that the character set of content should be translated. If the filter chain is explicitly configured using the `AddOutputFilter` directive, `NoImplicitAdd` should be specified so that `mod_charset_lite` doesn't add its filter.

### TranslateAllMimeTypes | NoTranslateAllMimeTypes

Normally, `mod_charset_lite` will only perform translation
on a small subset of possible mimetypes. When the TranslateAllMimeTypes keyword is specified for a given configuration section, translation is performed without regard for mimetype.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Source charset of files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td><code>CharsetSourceEnc charset</code></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host, directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>FileInfo</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td><code>mod_charset_lite</code></td>
</tr>
</tbody>
</table>

The `CharsetSourceEnc` directive specifies the source charset of files in the associated container.

The value of the `charset` argument must be accepted as a valid character set name by the character set support in [APR](https://httpd.apache.org/docs/2.4/modules/mod_charset_lite.html). Generally, this means that it must be supported by `iconv`.

### Example

```plaintext
<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 mod_dav

<table>
<thead>
<tr>
<th>:</th>
<th>(WebDAV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>Extension</td>
</tr>
<tr>
<td>:</td>
<td>dav_module</td>
</tr>
<tr>
<td>:</td>
<td>mod_dav.c</td>
</tr>
</tbody>
</table>

1 2

WebDAV ("")

DavLockDB
LimitXMLRequestBody
WebDAV Resources
Enabling WebDAV

mod_dav  httpd.conf:

Dav On

DAV DAV

DAV  DavLockDB

DAVLockDB /usr/local/apache2/var/DavLock

Apache

<Limit>  <Location> DAV

LimitXMLRequestBody

DavLockDB /usr/local/apache2/var/DavLock

<Location /foo>
  Order Allow,Deny
  Allow from all
  Dav On

  AuthType Basic
  AuthName DAV
  AuthUserFile user.passwd

  <LimitExcept GET OPTIONS>
    require user admin
  </LimitExcept>
</Location>
DAV

DAV
WebDAV

mod_dav
Group

mod_dav

DavDepthInfinity

DAV

HTT
SSL

Apache

Limit XMLRequestBody

PROPFIND
<table>
<thead>
<tr>
<th>mod_dav</th>
<th>(PHP CGI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>DAV</td>
</tr>
</tbody>
</table>

| Alias /phparea /home/gstein/php_files |
| Alias /php-source /home/gstein/php_files |
| <Location /php-source>                 |
|     DAV On                             |
|     ForceType text/plain               |
| </Location>                           |

http://example.com/phparea PHP
http://example.com/php-source DAV
| :   | WebDAV HTTP       | Dav On|Off|provider-name |
| :   | Dav Off          |

| :   | Extension         | mod_dav |

**WebDAV HTTP**

```
<Location /foo>
  Dav On
</Location>
```

**On**  

```
  mod_dav_fs
```

**filesystem**  

```
WebDAV
```
PROPFIND, Depth: Infinity
DavDepthInfinity on|off
DavDepthInfinity off
Extension
mod_dav

'Depth: Infinity' PROPFIND
denial-of-service
DAV

DavMinTimeout  seconds

DavMinTimeout 0

Extension

mod_dav

DAV

DavMin

(600)

<Location /MSWord>
  DavMinTimeout 600
</Location>
Apache mod_dav_fs

mod_dav

Extension
dav_fs_module
mod_dav_fs.c

mod_dav

Dav filesystem

filesystem mod_dav

mod_dav
DavLockDB:

```
DAV
DavLockDB file-path
, Extension
mod_dav_fs
```

SDBM

DavLockDB logs/DavLock
Apache mod_dav_lock

mod_dav
Extension
dav_lock_module
mod_dav_lock.c
2.1

mod_dav  API
subversion  mod_dav_svn
mod_dav_fs
mod_dav_lock  DavGenericLockDB
ap_lookup_provider

mod_dav

mod_dav
<table>
<thead>
<tr>
<th></th>
<th>DAV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DavGenericLockDB file-path</td>
</tr>
<tr>
<td></td>
<td>,</td>
</tr>
<tr>
<td></td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td>mod_dav_lock</td>
</tr>
</tbody>
</table>

DavGenericLockDB

| mod_dav_lock | SDBM                      |

DavGenericLockDB var/DavLock

<table>
<thead>
<tr>
<th>Apache</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerRoot</td>
<td>var/</td>
</tr>
</tbody>
</table>

Copyright 2017 The Apache Software Foundation.

[Apache License, Version 2.0](https://www.apache.org/licenses/LICENSE-2.0)
Apache Module mod_dbd

<table>
<thead>
<tr>
<th>Description:</th>
<th>Manages SQL database connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>Extension</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>dbd_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mod_dbd.c</td>
</tr>
<tr>
<td>Compatibility:</td>
<td>Version 2.1 and later</td>
</tr>
</tbody>
</table>

Summary

**mod_dbd** manages SQL database connections using [APR](https://apr.apache.org). It provides database connections on request to modules requiring SQL database functions, and takes care of managing databases with optimal efficiency and scalability for both threaded and non-threaded MPMs. For details, see the [APR](https://apr.apache.org) website and this overview of the Apache DBD Framework by its original developer.

See also

- [Password Formats](https://httpd.apache.org/docs/2.4/ssl/password-formats.html)
This module manages database connections, in a manner optimised for the platform. On non-threaded platforms, it provides a persistent connection in the manner of classic LAMP (Linux, Apache, Mysql, Perl/PHP/Python). On threaded platform, it provides an altogether more scalable and efficient connection pool, as described in this article at ApacheTutor. Note that mod_dbd supersedes the modules presented in that article.
mod_dbd exports five functions for other modules to use. The API is as follows:

typedef struct {
    apr_dbd_t *handle;
    apr_dbd_driver_t *driver;
    apr_hash_t *prepared;
} ap_dbd_t;

/* Export functions to access the database */

/* acquire a connection that MUST be explicitly closed. * Returns NULL on error */
AP_DECLARE(ap_dbd_t*) ap_dbd_open(apr_pool_t*, server_rec*);

/* release a connection acquired with ap_dbd_open */
AP_DECLARE(void) ap_dbd_close(server_rec*, ap_dbd_t*);

/* acquire a connection that will have the lifetime of a request * and MUST NOT be explicitly closed. Return NULL on error. * This is the preferred function for most applications. */
AP_DECLARE(ap_dbd_t*) ap_dbd_acquire(request_rec*);

/* acquire a connection that will have the lifetime of a connection * and MUST NOT be explicitly closed. Return NULL on error. */
AP_DECLARE(ap_dbd_t*) ap_dbd_cacquire(conn_rec*);

/* Prepare a statement for use by a client module */
AP_DECLARE(void) ap_dbd_prepare(server_rec*, const char*, const char*);

/* Also export them as optional functions for modules that prefer */
APR_DECLARE_OPTIONAL_FN(ap_dbd_t*, ap_dbd_open, (apr_pool_t*, server_rec*), ap_dbd_close, (server_rec*, ap_dbd_t*), ap_dbd_acquire, (request_rec*), ap_dbd_cacquire, (conn_rec*))};
APR_DECLARE_OPTIONAL_FN(void, ap_dbd_prepare, (server_rec*, const char*, const char*))};
mod_dbd supports SQL prepared statements on behalf of modules that may wish to use them. Each prepared statement must be assigned a name (label), and they are stored in a hash: the prepared field of an ap_dbd_t. Hash entries are of type apr_dbd_prepared_t and can be used in any of the apr_dbd prepared statement SQL query or select commands.

It is up to dbd user modules to use the prepared statements and document what statements can be specified in httpd.conf, or to provide their own directives and use ap_dbd_prepare.

Caveat

When using prepared statements with a MySQL database, it is preferred to set reconnect to 0 in the connection string as to avoid errors that arise from the MySQL client reconnecting without properly resetting the prepared statements. If set to 1, any broken connections will be attempted fixed, but as mod_dbd is not informed, the prepared statements will be invalidated.
Any web/database application needs to secure itself against SQL injection attacks. In most cases, Apache DBD is safe, because applications use prepared statements, and untrusted inputs are only ever used as data. Of course, if you use it via third-party modules, you should ascertain what precautions they may require.

However, the FreeTDS driver is inherently unsafe. The underlying library doesn't support prepared statements, so the driver emulates them, and the untrusted input is merged into the SQL statement.

It can be made safe by untainting all inputs: a process inspired by Perl's taint checking. Each input is matched against a regexp, and only the match is used, according to the Perl idiom:

```
$untrusted =~ /([a-z]+)/;
$trusted = $1;
```

To use this, the untainting regexps must be included in the prepared statements configured. The regexp follows immediately after the % in the prepared statement, and is enclosed in curly brackets {}. For example, if your application expects alphanumeric input, you can use:

```
"SELECT foo FROM bar WHERE input = %s"
```

with other drivers, and suffer nothing worse than a failed query. But with FreeTDS you'd need:

```
"SELECT foo FROM bar WHERE input = %{{[A-Za-z0-9]+}}s"
```

Now anything that doesn't match the regexp's $1 match is discarded, so the statement is safe.
An alternative to this may be the third-party ODBC driver, which offers the security of genuine prepared statements.
**Description:** Keepalive time for idle connections

**Syntax:** DBDExptime *time-in-seconds*

**Default:** DBDExptime 300

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_dbd

Set the time to keep idle connections alive when the number of connections specified in DBDKeep has been exceeded (threaded platforms only).
**Description:** Maximum sustained number of connections

**Syntax:** DBDKeep *number*

**Default:** DBDKeep 2

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_dbd

Set the maximum number of connections per process to be sustained, other than for handling peak demand (threaded platforms only).
**DBDMax**

**Description:** Maximum number of connections

**Syntax:** `DBDMax number`

**Default:** `DBDMax 10`

**Context:** server config, virtual host

**Status:** Extension

**Module:** `mod_dbd`

Set the hard maximum number of connections per process (threaded platforms only).
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Minimum number of connections</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>DBDMin <em>number</em></td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>DBDMin 1</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_dbd</td>
</tr>
</tbody>
</table>

Set the minimum number of connections per process (threaded platforms only).
**Description:** Parameters for database connection

**Syntax:**

```
DBDParams
param1=value1[,param2=value2]
```

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_dbd

As required by the underlying driver. Typically this will be used to pass whatever cannot be defaulted amongst username, password, database name, hostname and port number for connection.

Connection string parameters for current drivers include:

**FreeTDS (for MSSQL and SyBase - see SECURITY note)**

- username, password, appname, dbname, host, charset, lang, server

**MySQL**

- host, port, user, pass, dbname, sock, flags, fildsz, group, reconnect

**ODBC**

- datasource, user, password, connect, ctimeout, stimeout, access, txmode, bufsize

**Oracle**

- user, pass, dbname, server

**PostgreSQL**

- The connection string is passed straight through to PQconnectdb

**SQLite2**

- The connection string is split on a colon, and part1:part2 is used as sqlite_open(part1, atoi(part2), NULL)

**SQLite3**
The connection string is passed straight through to sqlite3_open
DBDPersist

**Description:** Whether to use persistent connections

**Syntax:** DBDPersist On|Off

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_dbd

If set to Off, persistent and pooled connections are disabled. A new database connection is opened when requested by a client, and closed immediately on release. This option is for debugging and low-usage servers.

The default is to enable a pool of persistent connections (or a single LAMP-style persistent connection in the case of a non-threaded server), and should almost always be used in operation.

Prior to version 2.2.2, this directive accepted only the values 0 and 1 instead of Off and On, respectively.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Define an SQL prepared statement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td><code>DBDPrepareSQL &quot;SQL statement&quot; label</code></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td><code>mod_dbd</code></td>
</tr>
</tbody>
</table>

For modules such as authentication that repeatedly use a single SQL statement, optimum performance is achieved by preparing the statement at startup rather than every time it is used. This directive prepares an SQL statement and assigns it a label.
**Description:** Specify an SQL driver

**Syntax:** DBDriver *name*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_dbd

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 mod_deflate

mod_deflate DEFLECT

Filters
AddOutputFilterByType DEFLATE text/html text/plain text/xml

<Location />
  # Insert filter
  SetOutputFilter DEFLATE

  # Netscape 4.x has some problems...
  BrowserMatch ^Mozilla/4 gzip-only-text/html

  # Netscape 4.06-4.08 have some more problems
  BrowserMatch ^Mozilla/4\.[0678] no-gzip

  # MSIE masquerades as Netscape, but it is fine
  BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
  # Don't compress images
  SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary

  # Make sure proxies don't deliver the wrong content
  Header append Vary User-Agent env=!dont-vary
</Location>
Output Compression

DEFLATE

SetOutputFilter DEFLATE

MIME

AddOutputFilterByType DEFLATE text/html

<Directory "/your-server-root/manual">
  AddOutputFilterByType DEFLATE text/html
</Directory>

BrowserMatch only-text/html

BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

User-Agent Netscape Navigator 4.x
  4.06, 4.07, 4.08 html

3 BrowserMatch "Mozilla/4"

DEFLATE PHP SSI RESOURCE
SetEnv force-gzip accept-encoding

mod_deflate gzip inflate/uncompress
   SetOutputFilter AddOutputFilter INFLATE

<Location /dav-area>
   ProxyPass http://example.com/
   SetOutputFilter INFLATE
</Location>

example.com gzip

mod_deflate gzip
   AddInputFilter DEFLATE

<Location /dav-area>
   SetInputFilter DEFLATE
</Location>

Content-Encoding: gzip

Content-Length
mod_deflate  Vary: Accept-Encoding HTTP
Accept-Encoding

User-Agent  Vary
DEFLATE

Header append Vary User-Agent

(HTTP)  Vary  *

Header set Vary *
<table>
<thead>
<tr>
<th>DeflateBufferSize value</th>
<th>zlib</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflateBufferSize 8096</td>
<td></td>
</tr>
<tr>
<td>,</td>
<td></td>
</tr>
<tr>
<td>Extension</td>
<td></td>
</tr>
<tr>
<td>mod_deflate</td>
<td></td>
</tr>
</tbody>
</table>

```bash
DeflateBufferSize zlib
```
DeflateCompressionLevel

1() 9()
DeflateFilterNote

DeflateFilterNote ratio

LogFormat "%r" %b (%{ratio}n) "%{User-agent}i" deflate
customLog logs/deflate_log deflate

**Input**

**Output**

**Ratio**

( / * 100 )

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
Maximum size of inflated request bodies

DeflateInflateLimitRequestBody

None, but LimitRequestBody applies after deflation

.htaccess

Extension

mod_deflate

2.2.28 and later
Maximum number of times the inflation ratio for request bodies can be crossed

**DeflateInflateRatioBurst** value

3

,**,.htaccess

Extension

mod_deflate

2.2.28 and later
<table>
<thead>
<tr>
<th>DeflateInflateRatioLimit value</th>
<th>Maximum inflation ratio for request bodies</th>
<th>200</th>
</tr>
</thead>
</table>

, , .htaccess
Extension
mod_deflate
2.2.28 and later
: zlib
: DeflateMemLevel value
: DeflateMemLevel 9
: ,
: Extension
: mod_deflate

DeflateMemLevel   zlib (1 9)  
)


deflateWindowSize

Zlib

DeflateWindowSize value

DeflateWindowSize 15

Extension

mod_deflate

DeflateWindowSize zlib (: zlib )

( : 2 )
Apache mod_dir

Base
dir_module
mod_dir.c

index.html

mod_dir

() 

http://servername/foo/dirname URL
http://servername/foo/dirname/
DirectoryIndex local-url [local-url] ...
DirectoryIndex index.html
.,.,.htaccess
Indexes
Base
mod_dir

URL

DirectoryIndex index.html

http://myserver/docs/
http://myserver/docs/index.html URL

DirectoryIndex index.html index.txt /cgi-bin/index.pl

index.html  index.txt CGI
DirectorySlash On|Off
DirectorySlash On
.,,.htaccess
Indexes
Base
mod_dir
2.0.51

URL

- URL
- mod_autoindex
- DirectoryIndex
- HTML URL

# see security warning below!
<Location /some/path>
  DirectorySlash Off
  SetHandler some-handler
</Location>

DirectoryIndex (index.html) URL
index.html
Define a default URL for requests that don't map to a file

FallbackResource disabled | local-url
None - httpd will return 404 (Not Found)
.,.,.htaccess
Indexes
Base
mod_dir
Apache HTTP Server 2.2.16 and later - The disabled argument is supported since 2.2.24
Apache mod_disk_cache

mod_disk_cache

URI

htcacheclean

mod_disk_cache  mod_cache
<table>
<thead>
<tr>
<th>CacheDirLength</th>
<th>length</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheDirLevels</td>
<td><em>CacheDirLength</em></td>
<td>20</td>
</tr>
<tr>
<td>CacheDirLength</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CacheDirLevels levels</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>CacheDirLevels 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extension</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mod_disk_cache</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
CacheDirLevels

CacheDirLevels* CacheDirLength 20

CacheDirLevels 5
```
CacheMaxFileSize

CacheMaxFileSize 64000
<table>
<thead>
<tr>
<th>CacheMinFileSize</th>
<th>bytes</th>
<th>CacheMinFileSize</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extension</td>
<td></td>
<td>mod_disk_cache</td>
<td></td>
</tr>
</tbody>
</table>
CacheRoot directory

Extension

mod_disk_cache

CacheRoot c:/cacheroot
Apache mod_dumpio

I/O
Extension
dumpio_module
mod_dumpio.c

mod_dumpio Apache

SSL () SSL

()
Apache
DumpIOInput On

DumpIOInput Off

Extension
mod_dumpio
DumpIOInput Apache 2.1.3

DumpIOInput On
Controls the logging level of the DumpIO output

DumpIOLogLevel level

DumpIOLogLevel debug

Extension

mod_dumpio

DumpIOLogLevel is only available in Apache 2.2.4 and later.
DumpIOOutput On|Off

DumpIOOutput Off

Extension
mod_dumpio
DumpIOOutput Apache 2.1.3

DumpIOOutput On
Apache mod_echo

Experimental

echo_module

mod_echo.c

Apache 2.0
ProtocolEcho

ProtocolEcho On
PassEnv env-variable [env-variable] ...

, , .htaccess
FileInfo
Base
mod_env

httpd  CGI

PassEnv LD_LIBRARY_PATH
SetEnv env-variable value

FileInfo
Base
mod_env

CGI  SSI

SetEnv SPECIAL_PATH /foo/bin
UnsetEnv env-variable [env-variable] ...

, , .htaccess
FileInfo
Base
mod_env

CGI  SSI

UnsetEnv LD_LIBRARY_PATH
Apache Module mod_example

<table>
<thead>
<tr>
<th>Description:</th>
<th>Illustrates the Apache module API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>Experimental</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>example_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mod_example.c</td>
</tr>
</tbody>
</table>

**Summary**

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.
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`, `LDFLAGS` 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:

```html
<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:

```html
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.
The **Example** directive just sets a demonstration flag which the example module's content handler displays. It takes no arguments. If you browse to an URL to which the example content-handler applies, you will get a display of the routines within the module and how and in what order they were called to service the document request. The effect of this directive one can observe under the point "Example directive declared here: YES/NO".
Expires HTTP Cache-Control max-age

max-age (RFC 2616 section 14.9) Cache-Control Header
ExpiresDefault  ExpiresByType  :

ExpiresDefault "<base> [plus] {<num> <type>}"  
ExpiresByType type/encoding "<base> [plus] {<num> <type>}"  

<base>:

- access
- now('access')
- modification

plus <num> ( atoi()) < t

- years
- months
- weeks
- days
- hours
- minutes
- seconds

1  :

ExpiresDefault "access plus 1 month"
ExpiresDefault "access plus 4 weeks"
ExpiresDefault "access plus 30 days"

'<num> <type>'  :

ExpiresByType text/html "access plus 1 month 15 days 2 hours"
ExpiresByType image/gif "modification plus 5 hours 3 minutes"

Expires
Expires Cache-Control

ExpiresDefault ()

Expires Cache-Control
## MIME Expires

ExpiresByType MIME-type <code>seconds</code>


text/html

### # mod_expires

ExpiresActive On

# expire GIF images after a month in the client's cache
ExpiresByType image/gif A2592000

# HTML documents are good for a week from the time they were changed
ExpiresByType text/html M604800

ExpiresActive On
ExpiresDefault <code>seconds</code>

Indexes

Extension

mod_expires
Apache mod_ext_filter

mod_ext_filter
Apache API  Apache
# 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>
```

: gzip

```
# 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>
```
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>

sed

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>

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>

:\n
#!/usr/local/bin/perl -w
use strict;

open(SAVE, ">$ARGV[0]")
  or die "can't open $ARGV[0]: $?";

while (<STDIN>) {
  print SAVE $_;
  print $_;#
}

close(SAVE);
ExtFilterDefine

*filtername* parameters

Extension

mod_ext_filter

ExtFilterDefine

*filtername*  

**cmd**=*cmdline*

    cmd=

        DOCUMENT_PATH_INFO, and
        QUERY_STRING_UNESCAPED

**mode**=*mode*

    mode=output ()
    Apache 2.1

**intype**=*imt*  

**outtype**=*imt*  

PreservesContentLength

PreservesContentLength  

content length

**ftype**=*filtertype*  

AP_FTYPE_RE

AP_FTYPE_foo
disableenv=env

enableenv=env
mod_ext_filter

ExtFilterOptions option [option] ...
ExtFilterOptions DebugLevel=0 NoLogStderr

Extension
mod_ext_filter

ExtFilterOptions mod_ext_filter

DebugLevel=n
  DebugLevel mod_ext_filter
  DebugLevel=0
  DBGLVL_
  : Apache core

LogStderr | NoLogStderr
  LogStderr

ExtFilterOptions LogStderr DebugLevel=0

Apache
Apache Module mod_file_cache

**Description:** Caches a static list of files in memory

**Status:** Experimental

**Module Identifier:** file_cache_module

**Source File:** mod_file_cache.c

**Summary**

This module should be used with care. You can easily create a broken site using mod_file_cache, so read this document carefully.

*Caching* frequently requested files that change very infrequently is a technique for reducing server load. mod_file_cache provides two techniques for caching frequently requested static files. Through configuration directives, you can direct mod_file_cache to either open then mmap( ) a file, or to pre-open a file and save the file's open file handle. Both techniques reduce server load when processing requests for these files by doing part of the work (specifically, the file I/O) for serving the file when the server is started rather than during each request.

Notice: You cannot use this for speeding up CGI programs or other files which are served by special content handlers. It can only be used for regular files which are usually served by the Apache core content handler.

This module is an extension of and borrows heavily from the mod_mmap_static module in Apache 1.3.
**mod_file_cache** caches a list of statically configured files via **MMapFile** or **CacheFile** directives in the main server configuration.

Not all platforms support both directives. You will receive an error message in the server error log if you attempt to use an unsupported directive. If given an unsupported directive, the server will start but the file will not be cached. On platforms that support both directives, you should experiment with both to see which works best for you.

**MMapFile Directive**

The **MMapFile** directive of **mod_file_cache** maps a list of statically configured files into memory through the system call mmap(). This system call is available on most modern Unix derivates, but not on all. There are sometimes system-specific limits on the size and number of files that can be mmap()ed, experimentation is probably the easiest way to find out.

This mmap()ing is done once at server start or restart, only. So whenever one of the mapped files changes on the filesystem you have to restart the server (see the [Stopping and Restarting documentation](#)). To reiterate that point: if the files are modified in place without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as rdist and mv do this. The reason why this modules doesn't take care of changes to the files is that this check would need an extra stat() every time which is a waste and against the intent of I/O reduction.

**CacheFile Directive**
The **CacheFile** directive of **mod_file_cache** opens an active *handle* or *file descriptor* to the file (or files) listed in the configuration directive and places these open file handles in the cache. When the file is requested, the server retrieves the handle from the cache and passes it to the `sendfile()` (or `TransmitFile()` on Windows), socket API.

This file handle caching is done once at server start or restart, only. So whenever one of the cached files changes on the filesystem you *have* to restart the server (see the [Stopping and Restarting](#) documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` and `mv` do this.

**Note**

Don't bother asking for a directive which recursively caches all the files in a directory. Try this instead... See the [Include](#) directive, and consider this command:

```
find /www/htdocs -type f -print \
| sed -e 's/.*/mmapfile &/' > /www/conf/mmap.conf
```
**Description:** Cache a list of file handles at startup time

**Syntax:** CacheFile file-path [file-path] ...

**Context:** server config

**Status:** Experimental

**Module:** mod_file_cache

The **CacheFile** directive opens handles to one or more files (given as whitespace separated arguments) and places these handles into the cache at server startup time. Handles to cached files are automatically closed on a server shutdown. When the files have changed on the filesystem, the server should be restarted to re-cache them.

Be careful with the **file-path** arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links etc. because that again would cost extra stat() system calls which is not acceptable. This module may or may not work with filenames rewritten by **mod_alias** or **mod_rewrite**.

**Example**

CacheFile /usr/local/apache/htdocs/index.html
### MMapFile Directive

**Description:** Map a list of files into memory at startup time

**Syntax:** `MMapFile file-path [file-path] ...`

**Context:** server config

**Status:** Experimental

**Module:** `mod_file_cache`

The MMapFile directive maps one or more files (given as whitespace separated arguments) into memory at server startup time. They are automatically unmapped on a server shutdown. When the files have changed on the filesystem at least a HUP or USR1 signal should be send to the server to re-mmap() them.

Be careful with the `file-path` arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links etc. because that again would cost extra `stat()` system calls which is not acceptable. This module may or may not work with filenames rewritten by `mod_alias` or `mod_rewrite`.

**Example**

```
MMapFile /usr/local/apache/htdocs/index.html
```
Apache Module mod_filter

**Description:** Context-sensitive smart filter configuration module

**Status:** Base

**Module Identifier:** filter_module

**Source File:** mod_filter.c

**Compatibility:** Version 2.1 and later

**Summary**

This module enables smart, context-sensitive configuration of output content filters. For example, apache can be configured to process different content-types through different filters, even when the content-type is not known in advance (e.g. in a proxy).

**mod_filter** works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to **mod_filter**; no change to existing filter modules is required (although it may be possible to simplify them).
In the traditional filtering model, filters are inserted unconditionally using `AddOutputFilter` and family. Each filter then needs to determine whether to run, and there is little flexibility available for server admins to allow the chain to be configured dynamically.

`mod_filter` by contrast gives server administrators a great deal of flexibility in configuring the filter chain. In fact, filters can be inserted based on any Request Header, Response Header or Environment Variable. This generalises the limited flexibility offered by `AddOutputFilterByType`, and fixes it to work correctly with dynamic content, regardless of the content generator. The ability to dispatch based on Environment Variables offers the full flexibility of configuration with `mod_rewrite` to anyone who needs it.
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 1:** The traditional filter model

**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.
mod_filter normally only runs filters on responses with HTTP status 200 (OK). If you want to filter documents with other response statuses, you can set the `filter-errordocs` environment variable, and it will work on all responses regardless of status. To refine this further, you can use expression conditions with `FilterProvider`.
Server side Includes (SSI)

A simple case of using mod_filter in place of AddOutputFilterByType

```
FilterDeclare SSI
FilterProvider SSI INCLUDES resp=Content-Type $text/html
FilterChain SSI
```

Server side Includes (SSI)

The same as the above but dispatching on handler (classic SSI behaviour; .shtml files get processed).

```
FilterProvider SSI INCLUDES Handler server-parsed
FilterChain SSI
```

Emulating mod_gzip with mod_deflate

Insert INFLATE filter only if "gzip" is NOT in the Accept-Encoding header. This filter runs with 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>
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 getbyteranges 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.
**Description:** Configure the filter chain

**Syntax:** FilterChain [+=-@!]filter-name ...

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

**Override:** Options

**Status:** Base

**Module:** 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**
This directive declares an output filter together with a header or environment variable that will determine runtime configuration. The first argument is a filter-name for use in FilterProvider, FilterChain and FilterProtocol directives.

The final (optional) argument is the type of filter, and takes values of ap_filter_type - namely RESOURCE (the default), CONTENT_SET, PROTOCOL, TRANSCODE, CONNECTION or NETWORK.
**Description:** Deal with correct HTTP protocol handling

**Syntax:**
```
FilterProtocol filter-name [provider-name] proto-flags
```

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

**Override:** Options

**Status:** Base

**Module:** 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 onbyteranges 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 uncachable (eg by introducing randomised content changes)
This directive registers a provider for the smart filter. The provider will be called if and only if the match declared here matches the value of the header or environment variable declared as dispatch.

provider-name must have been registered by loading a module that registers the name with ap_register_output_filter.

The dispatch argument is a string with optional req=, resp= or env= prefix causing it to dispatch on (respectively) the request header, response header, or environment variable named. In the absence of a prefix, it defaults to a response header. A special case is the word handler, which causes mod_filter to dispatch on the content handler.

The match argument specifies a match that will be applied to the filter's dispatch criterion. The match may be a string match (exact match or substring), a regex, an integer (greater, less than or equals), or unconditional. The first characters of the match argument determines this:

First, if the first character is an exclamation mark (!), this reverses the rule, so the provider will be used if and only if the match fails.

Second, it interprets the first character excluding any leading ! as follows:
<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(none)</td>
<td>exact match</td>
</tr>
<tr>
<td>$</td>
<td>substring match</td>
</tr>
<tr>
<td>/</td>
<td>regex match (delimited by a second /)</td>
</tr>
<tr>
<td>=</td>
<td>integer equality</td>
</tr>
<tr>
<td>&lt;</td>
<td>integer less-than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>integer less-than or equal</td>
</tr>
<tr>
<td>&gt;</td>
<td>integer greater-than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>integer greater-than or equal</td>
</tr>
<tr>
<td>*</td>
<td>Unconditional match</td>
</tr>
</tbody>
</table>
**Description:** Get debug/diagnostic information from `mod_filter`

**Syntax:** `FilterTrace filter-name level`

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

**Status:** Base

**Module:** `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 mod_headers

HTTP

Extension
headers_module
mod_headers.c
Mod_header

RequestHeader append MirrorID "mirror 12"
RequestHeader unset MirrorID

MirrorID

MirrorID "mirror 12"
mod_headers

/  early

early
1. "TS"

Header echo ^TS

2. MyHeader

Header add MyHeader "%D %t"

MyHeader: D=3775428 t=991424704447256

3. Joe:

Header add MyHeader "Hello Joe. It took %D microseconds for Apache to serve this request."

MyHeader: Hello Joe. It took D=3775428 microseconds for Apache to serve this request.

4. "MyRequestHeader" MyHeader

mod_setenvif

SetEnvIf MyRequestHeader value HAVE_MyRequestHeader
Header add MyHeader "%D %t mytext" env=HAVE_MyRequestHeader

HTTP MyRequestHeader: value

MyHeader: D=3775428 t=991424704447256 mytext
### HTTP

<table>
<thead>
<tr>
<th>Condition</th>
<th>Append</th>
<th>Add</th>
<th>Unset</th>
<th>Echo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header [condition]</td>
<td>set</td>
<td>append</td>
<td>add</td>
<td>unset</td>
</tr>
<tr>
<td>header [value]</td>
<td>[early</td>
<td>env=[!]variable]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### .htaccess

- FileInfo
- Extension
- mod_headers

```plaintext
HTTP

condition  onsuccess  always
always (2xx) : 

set

append

add ()

unset

echo 

( header)

header

add, append, set value value
value : 
```
<table>
<thead>
<tr>
<th>%s 2.1</th>
<th>SSL0ptions +StdEnvVars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>env=!... )</td>
</tr>
<tr>
<td></td>
<td>Header</td>
</tr>
<tr>
<td></td>
<td>Header</td>
</tr>
</tbody>
</table>
HTTP

set

append

add

unset

( header)
  value                unset  \v
Header

RequestHeader
  env=!... )      RequestHeader

   early         RequestHeader   fixup
Apache mod_ident

- RFC 1413 ident lookups
- Extension
- ident_module
- mod_ident.c
- Apache 2.1

RFC 1413

mod log config
| RFC 1413 | IdentityCheck  On|Off |
|----------|------------------|
|          | IdentityCheck  Off |
|          | ,               |
|          | Extension       |
|          | mod_ident       |
|          | Apache 2.1 core |

identd
<table>
<thead>
<tr>
<th>Ident</th>
<th>IdentityCheckTimeout seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IdentityCheckTimeout 30</td>
</tr>
</tbody>
</table>

Extension: mod_ident

ident 30
Apache Module mod_imagemap

<table>
<thead>
<tr>
<th>Description:</th>
<th>Server-side imagemap processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>Base</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>imagemap_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mod_imagemap.c</td>
</tr>
</tbody>
</table>

Summary

This module processes .map files, thereby replacing the functionality of the imagemap CGI program. Any directory or document type configured to use the handler imap-file (using either AddHandler or SetHandler) will be processed by this module.

The following directive will activate files ending with .map as imagemap files:

AddHandler imap-file map

Note that the following is still supported:

AddType application/x-httpd-imap map

However, we are trying to phase out "magic MIME types" so we are deprecating this method.
The imagemap module adds some new features that were not possible with previously distributed imagemap programs.

- URL references relative to the Referer: information.
- Default `<base>` assignment through a new map directive `base`.
- No need for `imagemap.conf` file.
- Point references.
- Configurable generation of imagemap menus.
The lines in the imagemap files can have one of several formats:

```
directive value [x,y ...]
directive value "Menu text" [x,y ...]
directive value x,y ... "Menu text"
```

The directive is one of base, default, poly, circle, rect, or point. The value is an absolute or relative URL, or one of the special values listed below. The coordinates are $x,y$ pairs separated by whitespace. The quoted text is used as the text of the link if a imagemap menu is generated. Lines beginning with '#' are comments.

**Imagemap File Directives**

There are six directives allowed in the imagemap file. The directives can come in any order, but are processed in the order they are found in the imagemap file.

**base Directive**

Has the effect of `<base href="value">`. The non-absolute URLs of the map-file are taken relative to this value. The base directive overrides `ImapBase` as set in a `.htaccess` file or in the server configuration files. In the absence of an `ImapBase` configuration directive, base defaults to `http://server_name/`.

`base_uri` is synonymous with `base`. Note that a trailing slash on the URL is significant.

**default Directive**

The action taken if the coordinates given do not fit any of the poly, circle or rect directives, and there are no point directives. Defaults to `nocontent` in the absence of an
**ImapDefault** configuration setting, causing a status code of 204 No Content to be returned. The client should keep the same page displayed.

**poly Directive**
Takes three to one-hundred points, and is obeyed if the user selected coordinates fall within the polygon defined by these points.

**circle**
Takes the center coordinates of a circle and a point on the circle. Is obeyed if the user selected point is within the circle.

**rect Directive**
Takes the coordinates of two opposing corners of a rectangle. Is 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 be 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 be 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:

```html
<a href="http://example.com/">Menu text</a>
```

If no quoted text is present, the name of the link will be used
as the text:

```html
<a href="http://example.com/">http://example.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.example.com/lincoln/feedback/ 195,0
305,27
rect another_file "in same directory as referer" 306,0 419,27
point http://www.zyzzyva.example.com/ 100,100
point http://www.tripod.example.com/ 200,200
rect mailto:nate@tripod.example.com 100,150 200,0 "Bugs?"
Referencing your mapfile

**HTML example**

```html
<a href="/maps/imagemap1.map">
    <img ismap src="/images/imagemap1.gif">
</a>
```

**XHTML example**

```html
<a href="/maps/imagemap1.map">
    <img ismap="ismap" src="/images/imagemap1.gif" />
</a>
```
**Description:** Default base for imagemap files

**Syntax:**
ImapBase map|referer|URL

**Default:**
ImapBase http://servername/

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

**Override:**
Indexes

**Status:**
Base

**Module:**
mod_imagemap

The `ImapBase` directive sets the default base used in the imagemap files. Its value is overridden by a base directive within the imagemap file. If not present, the base defaults to `http://servername/`.

**See also**

- [UseCanonicalName](#)
The **ImapDefault** directive sets the default default used in the imagemap files. Its value is overridden by a default directive within the imagemap file. If not present, the default action is `nocontent`, which means that a 204 No Content is sent to the client. In this case, the client should continue to display the original page.
**Description:** Action if no coordinates are given when calling an imagemap

**Syntax:**
```
ImapMenu
none|formatted|semiformatted|unformatted
```

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

**Override:** Indexes

**Status:** Base

**Module:** mod_imagemap

The **ImapMenu** directive determines the action taken if an imagemap file is called without valid coordinates.

**none**
If ImapMenu is none, no menu is generated, and the default action is performed.

**formatted**
A formatted menu is the simplest menu. Comments in the imagemap file are ignored. A level one header is printed, then an hrule, then the links each on a separate line. The menu has a consistent, plain look close to that of a directory listing.

**semiformatted**
In the semiformatted menu, comments are printed where they occur in the imagemap file. Blank lines are turned into HTML breaks. No header or hrule is printed, but otherwise the menu is the same as a formatted menu.

**unformatted**
Comments are printed, blank lines are ignored. Nothing is printed that does not appear in the imagemap file. All breaks and headers must be included as comments in the imagemap file. This gives you the most flexibility over the appearance of your menus, but requires you to treat your map files as HTML
instead of plaintext.
Apache `mod_include`

- html (Server Side Includes)
- Base
- include_module
- mod_include.c
- Apache 2.0

Options
- AcceptPathInfo

SSI
Server Side Includes

INCLUDES

Server-side

.shtml  Apache

AddType text/html .shtml
AddOutputFilter INCLUDES .shtml

.shtml
Options .htaccess):

Options +Includes

server-parsed INCLUDES MIME
server-parsed-html text/x-server-parsed-html3
Apache INCLUDES (MIME

Server Side Includes
SSI  PATH_INFO ()
<!--#element attribute=value attribute=value ... -->

(: value) (') ()

( - - >) SSI

(: element)

<table>
<thead>
<tr>
<th>config</th>
<th>configure output formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>echo</td>
<td>print variables</td>
</tr>
<tr>
<td>exec</td>
<td>execute external programs</td>
</tr>
<tr>
<td>fsize</td>
<td>print size of a file</td>
</tr>
<tr>
<td>flastmod</td>
<td>print last modification time of a file</td>
</tr>
<tr>
<td>include</td>
<td>include a file</td>
</tr>
<tr>
<td>printenv</td>
<td>print all available variables</td>
</tr>
<tr>
<td>set</td>
<td>set a value of a variable</td>
</tr>
</tbody>
</table>

SSI  mod_include

config

echomsg (Apache 2.1)

echo

eerrmsg

sizefmt

)
timefmt
   strftime(3)

\textbf{echo}

\textbf{include} \underline{SSIUndefinedEcho}

\textbf{var}

\textbf{encoding}

\textbf{echo} \quad \textbf{entity} \quad \textbf{encoding}

\textbf{encoding} \quad \textbf{var} \quad \textbf{ISO-8859-1}

\textbf{exec}

exec CGI

\textbf{cgi}

\%-% URL (\texttt{ScriptAlias Options ExecCGI}) CGI

CGI \quad \text{PATH\_INFO} (\underline{CGI} \include)

<!--#exec cgi="/cgi-bin/example.cgi" -->
Location: HTML ()

exec  cgi  include virtual CGI

cgi  include virtual

<!--#include virtual="/cgi-bin/example.cgi?argument=value"-->

```bash
cmd
/bin/sh  CGI

#include virtual  #exec  cgi  #exec  cmd
(  #include virtual) Apache

Win32  suexec unix  exec
unix  suexec  Win32  suexec  unix:

<!--#exec  cmd="perl /path/to/perlscript arg1 arg2"-->
```

filesize

```bash
sizefmt
```

file

```bash
virtual
(% ) URL-path (/)
```

flastmod

```bash
timefmt
```

include
include

file

virtual

<!--#include virtual="/cgi-bin/example.cgi?argument=value"-->

printenv

Apache 1.3.12

set

var

value
CGI   echo   if   elif,
SSI
echo, set

<!--#if expr="$a = \$test" -->

: 

<!--#set var="Zed" value="${REMOTE_HOST}_${REQUEST_METHOD}" -->

REMOTE_HOST "X" REQUEST_METHOD "Y" Zed "X_Y"

DOCUMENT_URI /foo/file.html "in foo"
/bar/file.html "in bar" "in neither"

<!--#if expr='"$DOCUMENT_URI" = "/foo/file.html"' -->
in foo
<!--#elif expr='"$DOCUMENT_URI" = "/bar/file.html"' -->
in bar
<!--#else -->
in neither
<!--#endif -->
if

elif else test_condition

endif if

test_condition:

string
    string
    string1 = string2
    string1 == string2
    string1 != string2
        string1 string2
        perl 5
        == =
        ( = ==)

string1 < string2
string1 <= string2
string1 > string2
string1 >= string2
    string1 string2
    ( strcmp(3)) "100"
"20"
(test_condition)
test_condition
!
test_condition
test_condition

test_condition1 && test_condition2
test_condition1 test_condition2

test_condition1 || test_condition2
test_condition1 test_condition2

"==" "!=" "&&" "!!!"

<!-#if expr="$a = test1 && $b = test2" -->
<!-#if expr="($a = test1) && ($b = test2)" -->

&& ||

:

string1 string2 string1 string2

' string1 string2' string1 string2

•
• ( && ||)
•

-DDEBUG_INCLUDE
Enable the -A flag during conditional flow control processing.
SSIEnableAccess on|off
SSIEnableAccess off
.htaccess
Base
mod_include
<table>
<thead>
<tr>
<th>include</th>
<th>SSIEndTag</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSIEndTag</td>
<td>&quot;--&gt;&quot;,</td>
</tr>
<tr>
<td></td>
<td>Base</td>
<td>mod_include</td>
</tr>
<tr>
<td></td>
<td>2.0.30</td>
<td></td>
</tr>
</tbody>
</table>

- **mod include** include

SSIEndTag "%>"
SSIErrorMsg

mod_include

<!--#config errmsg=message -->

SSIErrorMsg "<!-- Error -->"
Controls whether ETags are generated by the server.

SSIETag on

SSIETag off

.htaccess

Base

mod_include

Available in version 2.2.15 and later.
Controls whether Last-Modified headers are generated by the server.

**SSILastModified** on|off

**SSILastModified** off

, .htaccess

Base

mod_include

Available in version 2.2.15 and later.
mod_include include

()
SSITimeFormat

SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"

,..., .htaccess

All

Base

mod_include

2.0.30

DATE echo

<!--#config timefmt=formatstring -->

SSITimeFormat "%R, %B %d, %Y"

"22:26, June 14, 2002"
<table>
<thead>
<tr>
<th>echo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIUndefinedEcho string</td>
</tr>
<tr>
<td>SSIUndefinedEcho &quot;(none)&quot;</td>
</tr>
<tr>
<td>..., .htaccess</td>
</tr>
<tr>
<td>All</td>
</tr>
<tr>
<td>Base</td>
</tr>
<tr>
<td>mod_include</td>
</tr>
<tr>
<td>2.0.34</td>
</tr>
</tbody>
</table>

"echo"  

mod

SSIUndefinedEcho "<!-- undef -->"
SSI
XBitHack on|off|full
XBitHack off
,,.htaccess
Options
Base
mod_include

XBitHack  HTML  MIME

off

on  
text/html html

full
  on

CGI  #include
Apache mod_info

mod_info

httpd.conf

<Location /server-info>
SetHandler server-info
</Location>

<Location /
SetHandler server-info
Order deny,allow
Deny from all
Allow from yourcompany.com
</Location>

http://your.host.example.com/server-info
mod_info

Apache

mod_authz_host

<Location /server-info>
    SetHandler server-info
    Order allow,deny
    # Allow access from server itself
    Allow from 127.0.0.1
    # Additionally, allow access from local workstation
    Allow from 192.168.1.17
</Location>
server-info
http://your.host.example.com/server-info?config

?<module-name>

?config

?hooks

?list

?server
mod_info

- Include, <IfModule>, <IfDefine>,
- ()
- .htaccess
- <Directory>
- mod_perl
AddModuleInfo module-name string

string module-name HTML :

Apache Module mod_isapi

**Description:** ISAPI Extensions within Apache for Windows

**Status:** Base

**Module Identifier:** isapi_module

**Source File:** mod_isapi.c

**Compatibility:** Win32 only

**Summary**

This module implements the Internet Server extension API. It allows Internet Server extensions (e.g. ISAPI .dll modules) to be served by Apache for Windows, subject to the noted restrictions.

ISAPI extension modules (.dll files) are written by third parties. The Apache Group does not author these modules, so we provide no support for them. Please contact the ISAPI's author directly if you are experiencing problems running their ISAPI extension. **Please do not post such problems to Apache's lists or bug reporting pages.**
In the server configuration file, use the `AddHandler` directive to associate ISAPI files with the `isapi-handler` handler, and map it to them with their file extensions. To enable any .dll file to be processed as an ISAPI extension, edit the `httpd.conf` file and add the following line:

```
AddHandler isapi-handler .dll
```

In older versions of the Apache server, `isapi-isa` was the proper handler name, rather than `isapi-handler`. For compatibility, configurations may continue using `isapi-isa` through all versions of Apache prior to 2.3.0.

There is no capability within the Apache server to leave a requested module loaded. However, you may preload and keep a specific module loaded by using the following syntax in your `httpd.conf`:

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

Whether or not you have preloaded an ISAPI extension, all ISAPI extensions are governed by the same permissions and restrictions as CGI scripts. That is, `Options ExecCGI` must be set for the directory that contains the ISAPI .dll file.

Review the `Additional Notes` and the `Programmer's Journal` for additional details and clarification of the specific ISAPI support offered by `mod_isapi`. 


Apache's ISAPI implementation conforms to all of the ISAPI 2.0 specification, except for some "Microsoft-specific" extensions dealing with asynchronous I/O. Apache's I/O model does not allow asynchronous reading and writing in a manner that the ISAPI could access. If an ISA tries to access unsupported features, including async I/O, a message is placed in the error log to help with debugging. Since these messages can become a flood, the directive ISAPILogNotSupported Off exists to quiet this noise.

Some servers, like Microsoft IIS, load the ISAPI extension into the server and keep it loaded until memory usage is too high, or unless configuration options are specified. Apache currently loads and unloads the ISAPI extension each time it is requested, unless the ISAPICacheFile directive is specified. This is inefficient, but Apache's memory model makes this the most effective method. Many ISAPI modules are subtly incompatible with the Apache server, and unloading these modules helps to ensure the stability of the server.

Also, remember that while Apache supports ISAPI Extensions, it does not support ISAPI Filters. Support for filters may be added at a later date, but no support is planned at this time.
If you are programming Apache 2.0 `mod_isapi` modules, you must limit your calls to `ServerSupportFunction` to the following directives:

**HSE_REQ_SEND_URL_REDIRECT_RESP**
Redirect the user to another location.
This must be a fully qualified URL (e.g. `http://server/location`).

**HSE_REQ_SEND_URL**
Redirect the user to another location.
This cannot be a fully qualified URL, you are not allowed to pass the protocol or a server name (e.g. simply `/location`).
This redirection is handled by the server, not the browser.

**Warning**
In their recent documentation, Microsoft appears to have abandoned the distinction between the two `HSE_REQ_SEND_URL` functions. Apache continues to treat them as two distinct functions with different requirements and behaviors.

**HSE_REQ_SEND_RESPONSE_HEADER**
Apache accepts a response body following the header if it follows the blank line (two consecutive newlines) in the headers string argument. This body cannot contain NULLs, since the headers argument is NULL terminated.

**HSE_REQ_DONE_WITH_SESSION**
Apache considers this a no-op, since the session will be finished when the ISAPI returns from processing.

**HSE_REQ_MAP_URL_TO_PATH**
Apache will translate a virtual name to a physical name.
HSE_APPEND_LOG_PARAMETER

This logged message may be captured in any of the following logs:

- in the \"%{isapi-parameter}\" component in a CustomLog directive
- in the %q log component with the ISAPIAppendLogToQuery On directive
- in the error log with the ISAPIAppendLogToErrors On directive

The first option, the %{isapi-parameter}n component, is always available and preferred.

HSE_REQ_IS_KEEP_CONN

Will return the negotiated Keep-Alive status.

HSE_REQ_SEND_RESPONSE_HEADER_EX

Will behave as documented, although the fKeepConn flag is ignored.

HSE_REQ_IS_CONNECTED

Will report false if the request has been aborted.

Apache returns FALSE to any unsupported call to ServerSupportFunction, and sets the GetLastError value to ERROR_INVALID_PARAMETER.

ReadClient retrieves the request body exceeding the initial buffer (defined by ISAPIReadAheadBuffer). Based on the ISAPIReadAheadBuffer setting (number of bytes to buffer prior to calling the ISAPI handler) shorter requests are sent complete to the extension when it is invoked. If the request is longer, the ISAPI extension must use ReadClient to retrieve the remaining request body.
WriteClient is supported, but only with the HSE_IO_SYNC flag or no option flag (value of 0). Any other WriteClient request will be rejected with a return value of FALSE, and a GetLastError value of ERROR_INVALID_PARAMETER.

GetServerVariable is supported, although extended server variables do not exist (as defined by other servers.) All the usual Apache CGI environment variables are available from GetServerVariable, as well as the ALL_HTTP and ALL_RAW values.

Apache 2.0 mod_isapi supports additional features introduced in later versions of the ISAPI specification, as well as limited emulation of async I/O and the TransmitFile semantics. Apache also supports preloading ISAPI .dlls for performance, neither of which were not available under Apache 1.3 mod_isapi.
**Description:** Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log

**Syntax:** ISAPIAppendLogToErrors on|off

**Default:** ISAPIAppendLogToErrors off

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

**Override:** FileInfo

**Status:** Base

**Module:** mod_isapi

Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the server error log.
<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>ISAPIAppendLogToQuery on</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>ISAPIAppendLogToQuery on</td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>server config, virtual host, directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override</strong></td>
<td>FileInfo</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Base</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>mod_isapi</td>
</tr>
</tbody>
</table>

Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field (appended to the CustomLog %q component).
**Description:** ISAPI .dll files to be loaded at startup

**Syntax:**

```
ISAPICacheFile file-path [file-path]
...```

**Context:** server config, virtual host

**Status:** Base

**Module:** mod_isapi

Specifies a space-separated list of file names to be loaded when the Apache server is launched, and remain loaded until the server is shut down. This directive may be repeated for every ISAPI .dll file desired. The full path name of each file should be specified. If the path name is not absolute, it will be treated relative to `ServerRoot`. 
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Fake asynchronous support for ISAPI callbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>ISAPIFakeAsync on</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>ISAPIFakeAsync off</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host, directory, .htaccess</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>FileInfo</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>FileInfo</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_isapi</td>
</tr>
</tbody>
</table>

While set to on, asynchronous support for ISAPI callbacks is simulated.
**Description:** Log unsupported feature requests from ISAPI extensions

**Syntax:** ISAPILogNotSupported on|off

**Default:** ISAPILogNotSupported off

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

**Override:** FileInfo

**Status:** Base

**Module:** mod_isapi

Logs all requests for unsupported features from ISAPI extensions in the server error log. This may help administrators to track down problems. Once set to on and all desired ISAPI modules are functioning, it should be set back to off.
Description: Size of the Read Ahead Buffer sent to ISAPI extensions

Syntax: ISAPIReadAheadBuffer size

Default: ISAPIReadAheadBuffer 49152

Context: server config, virtual host, directory, .htaccess

Override: FileInfo

Status: Base

Module: mod_isapi

Defines the maximum size of the Read Ahead Buffer sent to ISAPI extensions when they are initially invoked. All remaining data must be retrieved using the ReadClient callback; some ISAPI extensions may not support the ReadClient function. Refer questions to the ISAPI extension's author.
### Apache Module mod_ldap

| **Description:** | LDAP connection pooling and result caching services for use by other LDAP modules |
| **Status:** | Extension |
| **Module Identifier:** | ldap_module |
| **Source File:** | util_ldap.c |
| **Compatibility:** | Available in version 2.0.41 and later |

### Summary

This module was created to improve the performance of websites relying on backend connections to LDAP servers. In addition to the functions provided by the standard LDAP libraries, this module adds an LDAP connection pool and an LDAP shared memory cache.

To enable this module, LDAP support must be compiled into apr-util. This is achieved by adding the `--with-ldap` flag to the `configure` script when building Apache.

SSL/TLS support is dependent on which LDAP toolkit has been linked to APR. As of this writing, APR-util supports: OpenLDAP SDK (2.x or later), Novell LDAP SDK, Mozilla LDAP SDK, native Solaris LDAP SDK (Mozilla based), native Microsoft LDAP SDK, or the iPlanet (Netscape) SDK. See the APR website for details.
The following is an example configuration that uses `mod_ldap` to increase the performance of HTTP Basic authentication provided by `mod_authnz_ldap`.

```
# Enable the LDAP connection pool and shared
# memory cache. Enable the LDAP cache status
# handler. Requires that mod_ldap and mod_authnz_ldap
# be loaded. Change the "yourdomain.example.com" to
# match your domain.

LDAPSharedCacheSize 500000
LDAPCacheEntries 1024
LDAPCacheTTL 600
LDAPOpCacheEntries 1024
LDAPOpCacheTTL 600

<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one
  AuthzLDAPAuthoritative off
  Require valid-user
</Location>
```
LDAP connections are pooled from request to request. This allows the LDAP server to remain connected and bound ready for the next request, without the need to unbind/connect/rebind. The performance advantages are similar to the effect of HTTP keepalives.

On a busy server it is possible that many requests will try and access the same LDAP server connection simultaneously. Where an LDAP connection is in use, Apache will create a new connection alongside the original one. This ensures that the connection pool does not become a bottleneck.

There is no need to manually enable connection pooling in the Apache configuration. Any module using this module for access to LDAP services will share the connection pool.
For improved performance, mod_ldap uses an aggressive caching strategy to minimize the number of times that the LDAP server must be contacted. Caching can easily double or triple the throughput of Apache when it is serving pages protected with mod_authnz_ldap. In addition, the load on the LDAP server will be significantly decreased.

mod_ldap supports two types of LDAP caching during the search/bind phase with a search/bind cache and during the compare phase with two operation caches. Each LDAP URL that is used by the server has its own set of these three caches.

**The Search/Bind Cache**

The process of doing a search and then a bind is the most time-consuming aspect of LDAP operation, especially if the directory is large. The search/bind cache is used to cache all searches that resulted in successful binds. Negative results (i.e., unsuccessful searches, or searches that did not result in a successful bind) are not cached. The rationale behind this decision is that connections with invalid credentials are only a tiny percentage of the total number of connections, so by not caching invalid credentials, the size of the cache is reduced.

mod_ldap stores the username, the DN retrieved, the password used to bind, and the time of the bind in the cache. Whenever a new connection is initiated with the same username, mod_ldap compares the password of the new connection with the password in the cache. If the passwords match, and if the cached entry is not too old, mod_ldap bypasses the search/bind phase.

The search and bind cache is controlled with the LDAPCacheEntries and LDAPCacheTTL directives.
**Operation Caches**

During attribute and distinguished name comparison functions, **mod_ldap** uses two operation caches to cache the compare operations. The first compare cache is used to cache the results of compares done to test for LDAP group membership. The second compare cache is used to cache the results of comparisons done between distinguished names.

The behavior of both of these caches is controlled with the `LDAPOpCacheEntries` and `LDAPOpCacheTTL` directives.

**Monitoring the Cache**

**mod_ldap** has a content handler that allows administrators to monitor the cache performance. The name of the content handler is `ldap-status`, so the following directives could be used to access the **mod_ldap** cache information:

```
<Location /server/cache-info>
  SetHandler ldap-status
</Location>
```

By fetching the URL `http://servername/cache-info`, the administrator can get a status report of every cache that is used by **mod_ldap** cache. Note that if Apache does not support shared memory, then each **httpd** instance has its own cache, so reloading the URL will result in different information each time, depending on which **httpd** instance processes the request.
The ability to create an SSL and TLS connections to an LDAP server is defined by the directives `LDAPTrustedGlobalCert`, `LDAPTrustedClientCert` and `LDAPTrustedMode`. These directives specify the CA and optional client certificates to be used, as well as the type of encryption to be used on the connection (none, SSL or TLS/STARTTLS).

```plaintext
# Establish an SSL LDAP connection on port 636. Requires that # mod_ldap and mod_authnz_ldap be loaded. Change the # "yourdomain.example.com" to match your domain.

LDAPTrustedGlobalCert CA_DER /certs/certfile.der

<Location /ldap-status>
   SetHandler ldap-status
   Order deny,allow
   Deny from all
   Allow from yourdomain.example.com
   AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
   AuthzLDAPAuthoritative off
   Require valid-user
</Location>

# Establish a TLS LDAP connection on port 389. Requires that # mod_ldap and mod_authnz_ldap be loaded. Change the # "yourdomain.example.com" to match your domain.

LDAPTrustedGlobalCert CA_DER /certs/certfile.der

<Location /ldap-status>
   SetHandler ldap-status
   Order deny,allow
   Deny from all
   Allow from yourdomain.example.com
   AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one TLS
   AuthzLDAPAuthoritative off
   Require valid-user
</Location>
```
The different LDAP SDKs have widely different methods of setting and handling both CA and client side certificates.

If you intend to use SSL or TLS, read this section CAREFULLY so as to understand the differences between configurations on the different LDAP toolkits supported.

**Netscape/Mozilla/iPlanet SDK**

CA certificates are specified within a file called cert7.db. The SDK will not talk to any LDAP server whose certificate was not signed by a CA specified in this file. If client certificates are required, an optional key3.db file may be specified with an optional password. The secmod file can be specified if required. These files are in the same format as used by the Netscape Communicator or Mozilla web browsers. The easiest way to obtain these files is to grab them from your browser installation.

Client certificates are specified per connection using the LDAPTrustedClientCert directive by referring to the certificate "nickname". An optional password may be specified to unlock the certificate's private key.

The SDK supports SSL only. An attempt to use STARTTLS will cause an error when an attempt is made to contact the LDAP server at runtime.

```
# Specify a Netscape CA certificate file
LDAPTrustedGlobalCert CA_CERT7_DB /certs/cert7.db
# Specify an optional key3.db file for client certificate support
LDAPTrustedGlobalCert CERT_KEY3_DB /certs/key3.db
# Specify the secmod file if required
LDAPTrustedGlobalCert CA_SECMOD /certs/secmod
<Location /ldap-status>
    SetHandler ldap-status
    Order deny,allow
    Deny from all
```
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.

```bash
# Specify two CA certificate files
LDAPTrustedGlobalCert CA_DER /certs/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
# Specify a client certificate file and key
LDAPTrustedGlobalCert CERT_BASE64 /certs/cert1.pem
LDAPTrustedGlobalCert KEY_BASE64 /certs/key1.pem [password]
# Do not use this directive, as it will throw an error
#LDAPTrustedClientCert CERT_BASE64 /certs/cert1.pem
```

OpenLDAP SDK

One or more CA certificates must be specified for the OpenLDAP SDK to work correctly. These certificates can be specified as binary DER or Base64 (PEM) encoded files.
Client certificates are specified per connection using the LDAPTrustedClientCert directive.

The documentation for the SDK claims to support both SSL and STARTTLS, however STARTTLS does not seem to work on all versions of the SDK. The SSL/TLS mode can be set using the LDAPTrustedMode parameter. If an ldaps:// URL is specified, SSL mode is forced. The OpenLDAP documentation notes that SSL (ldaps://) support has been deprecated to be replaced with TLS, although the SSL functionality still works.

```
# Specify two CA certificate files
LDAPTrustedGlobalCert CA_DER /certs/cacert1.der
LDAPTrustedGlobalCert CA_BASE64 /certs/cacert2.pem
<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  LDAPTrustedClientCert CERT_BASE64 /certs/cert1.pem
  LDAPTrustedClientCert KEY_BASE64 /certs/key1.pem
  AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
  AuthzLDAPAuthoritative off
  Require valid-user
</Location>
```

**Solaris SDK**

SSL/TLS for the native Solaris LDAP libraries is not yet supported. If required, install and use the OpenLDAP libraries instead.

**Microsoft SDK**

SSL/TLS certificate configuration for the native Microsoft LDAP libraries is done inside the system registry, and no configuration directives are required.

Both SSL and TLS are supported by using the ldaps:// URL format, or by using the LDAPTrustedMode directive accordingly.
Note: The status of support for client certificates is not yet known for this toolkit.
**Description:** Maximum number of entries in the primary LDAP cache

**Syntax:** LDAPCacheEntries *number*

**Default:** LDAPCacheEntries 1024

**Context:** server config

**Status:** Extension

**Module:** mod_ldap

Specifies the maximum size of the primary LDAP cache. This cache contains successful search/binds. Set it to 0 to turn off search/bind caching. The default size is 1024 cached searches.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Time that cached items remain valid</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>LDAPCacheTTL  \textit{seconds}</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>LDAPCacheTTL  600</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_ldap</td>
</tr>
</tbody>
</table>

Specifies the time (in seconds) that an item in the search/bind cache remains valid. The default is 600 seconds (10 minutes).
**Description**: Specifies the socket connection timeout in seconds

**Syntax**: LDAPConnectionTimeout *seconds*

**Context**: server config

**Status**: Extension

**Module**: mod_ldap

This directive configures the LDAP_OPT_NETWORK_TIMEOUT option in the underlying LDAP client library, when available. This value typically controls how long the LDAP client library will wait for the TCP connection to the LDAP server to complete.

If a connection is not successful with the timeout period, either an error will be returned or the LDAP client library will attempt to connect to a secondary LDAP server if one is specified (via a space-separated list of hostnames in the AuthLDAPURL).

The default is 10 seconds, if the LDAP client library linked with the server supports the LDAP_OPT_NETWORK_TIMEOUT option.

LDAPConnectionTimeout is only available when the LDAP client library linked with the server supports the LDAP_OPT_NETWORK_TIMEOUT option, and the ultimate behavior is dictated entirely by the LDAP client library.
| **Description:** | Number of entries used to cache LDAP compare operations |
| **Syntax:** | LDAPOpCacheEntries number |
| **Default:** | LDAPOpCacheEntries 1024 |
| **Context:** | server config |
| **Status:** | Extension |
| **Module:** | mod_ldap |

This specifies the number of entries mod_ldap will use to cache LDAP compare operations. The default is 1024 entries. Setting it to 0 disables operation caching.
**Description:** Time that entries in the operation cache remain valid

**Syntax:** LDAP0pCacheTTL  *seconds*

**Default:** LDAP0pCacheTTL  600

**Context:** server config

**Status:** Extension

**Module:** mod_ldap

Specifies the time (in seconds) that entries in the operation cache remain valid. The default is 600 seconds.
**Description:** Sets the shared memory cache file

**Syntax:** `LDAPSharedCacheFile directory-path/filename`

**Context:** server config

**Status:** Extension

**Module:** `mod_ldap`

Specifies the directory path and file name of the shared memory cache file. If not set, anonymous shared memory will be used if the platform supports it.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Size in bytes of the shared-memory cache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>LDAPSharedCacheSize  <em>bytes</em></td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>LDAPSharedCacheSize  500000</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_ldap</td>
</tr>
</tbody>
</table>

Specifies the number of bytes to allocate for the shared memory cache. The default is 500kb. If set to 0, shared memory caching will not be used.
**Description:** Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates.

**Syntax:** LDAPTrustedClientCert type directory-path/filename/nickname [password]

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

**Status:** Extension

**Module:** mod_ldap

It specifies the directory path, file name or nickname of a per connection client certificate used when establishing an SSL or TLS connection to an LDAP server. Different locations or directories may have their own independent client certificate settings. Some LDAP toolkits (notably Novell) do not support per connection client certificates, and will throw an error on LDAP server connection if you try to use this directive (Use the LDAPTrustedGlobalCert directive instead for Novell client certificates - See the SSL/TLS certificate guide above for details). The type specifies the kind of certificate parameter being set, depending on the LDAP toolkit being used. Supported types are:

- CERT_DER - binary DER encoded client certificate
- CERT_BASE64 - PEM encoded client certificate
- CERT_NICKNAME - Client certificate "nickname" (Netscape SDK)
- KEY_DER - binary DER encoded private key
- KEY_BASE64 - PEM encoded private key
**Description:** Sets the file or database containing global trusted Certificate Authority or global client certificates

**Syntax:** `LDAPTrustedGlobalCert type directory-path/filename [password]

**Context:** server config

**Status:** Extension

**Module:** 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
- CERT_PFX - PKCS#12 encoded client certificate (Novell SDK)
- KEY_DER - binary DER encoded private key
- KEY_BASE64 - PEM encoded private key
- KEY_PFX - PKCS#12 encoded private key (Novell SDK)
**Description:** Specifies the SSL/TLS mode to be used when connecting to an LDAP server.

**Syntax:** `LDAPTrustedMode type`

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ldap

The following modes are supported:

- NONE - no encryption
- SSL - ldaps:// encryption on default port 636
- TLS - STARTTLS encryption on default port 389

Not all LDAP toolkits support all the above modes. An error message will be logged at runtime if a mode is not supported, and the connection to the LDAP server will fail.

If an ldaps:// URL is specified, the mode becomes SSL and the setting of LDAPTrustedMode is ignored.
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Force server certificate verification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>LDAPVerifyServerCert  *On</td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>LDAPVerifyServerCert  <em>On</em></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_ldap</td>
</tr>
</tbody>
</table>

Specifies whether to force the verification of a server certificate when establishing an SSL connection to the LDAP server.
Apache mod_log_config

::
:: Base
:: log_config_module
:: mod_log_config.c

TransferLog CustomLog
<table>
<thead>
<tr>
<th>LogFormat</th>
<th>CustomLog</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; %&quot;</td>
<td>&quot;%&quot;</td>
</tr>
</tbody>
</table>

<p>|  %%       |            |
| %a       | IP         |
| %A       | IP         |
| %B       | HTTP       |
| %b       | HTTP CLF   |
| %        |            |
| {Foobar}C  | Foobar    |
| %D       |            |
| %        |            |
| {FOOBAR}e  | FOOBAR    |
| %f       |            |
| %h       |            |
| %H       |            |
| %        |            |
| {Foobar}i  | Foobar:    |
| %l       | (identd )  |
|          | IdentityCheck  |
|          | On |
| %m       |            |
| %        |            |
| {Foobar}n  | Foobar    |
| %        |            |
| {Foobar}o  | Foobar:    |
| %p       |            |
| %P       | ID         |
| %        |            |</p>
<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%q</code></td>
<td>( ? )</td>
</tr>
<tr>
<td><code>%r</code></td>
<td>---</td>
</tr>
<tr>
<td><code>%t</code></td>
<td>CLF ()</td>
</tr>
<tr>
<td><code>{format}t</code></td>
<td>format format strftime (3)</td>
</tr>
<tr>
<td><code>%T</code></td>
<td></td>
</tr>
<tr>
<td><code>%u</code></td>
<td>(( %s 401)</td>
</tr>
<tr>
<td><code>%U</code></td>
<td>URL</td>
</tr>
<tr>
<td><code>%v</code></td>
<td><strong>ServerName</strong></td>
</tr>
<tr>
<td><code>%V</code></td>
<td><strong>UseCanonicalName</strong></td>
</tr>
<tr>
<td><code>%X</code></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| |     X = 
| |     + = 
| |     - = 
| |     (Apache 1.3 %c |
| `%I` | 0 |
| `%O` | 0 |

"%" HTTP
400 500 User-agent
"%!200,304,302{Referer}i" 3
"<" ">"
2.0.46 %r, %i, %o
" \ C ( \n, \t ) 2.0.46

httpd 2.0 1.3 %b %B
)  mod_logio %0

:

Common Log Format (CLF)
"%h %l %u %t "%r" %>s %b"

Common Log Format
"%v %h %l %u %t "%r" %>s %b"

NCSA extended/combined
"%h %l %u %t "%r" %>s %b "%{Referer}i"
"%{User-agent}i"

Referer
"%{Referer}i -> %U"

Agent ()
"%{User-agent}i"
BufferedLogs

Base

mod_log_config

2.0.41

BufferedLogs

mod_log_config
CookieLog

filename

Base

mod_log_config

filename
CustomLog

file

ServerRoot

pipe

" |"

httpd

Unix

LogFormat

# CustomLog with format nickname
LogFormat "%h %l %u %t "%r" %>s %b" common
CustomLog logs/access_log common

# CustomLog with explicit format string
CustomLog logs/access_log "%h %l %u %t "%r" %>s %b"
mod_setenvif

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 format | nickname [nickname]
LogFormat "%h %l %u %t "%r" %>s %b"
Base
mod_log_config

LogFormat
LogFormat nickname
LogFormat format nickname
LogFormat CustomLog

nickname Nickname

LogFormat "%v %h %l %u %t "%r" %>s %b" vhost_common
TransferLog `file|pipe`

Base
mod_log_config

Log Format

```sh
LogFormat "%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-agent}i"
TransferLog logs/access_log
```
Apache mod_log_forensic

forensic

- Extension
- log_forensic_module
- mod_log_forensic.c
- mod_unique_id
  2.1

Forensic support check (Apache mod_log_config)
<table>
<thead>
<tr>
<th>forensic ID</th>
<th>Mod_unique_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>+yQtJf8CoAB4AAFNXBIEAAAAA</td>
<td>GET /manual/de/images/down.gif</td>
</tr>
<tr>
<td>HTTP/1.1</td>
<td>Host: localhost:%3a8080</td>
</tr>
<tr>
<td>ID:</td>
<td></td>
</tr>
<tr>
<td>-yQtJf8CoAB4AAFNXBIEAAAAA</td>
<td>check_forensic +/-</td>
</tr>
</tbody>
</table>
ForensicLog forensic
   mod_log_forensic  forensic-id
id}n transfer

2

filename
   ServerRoot

pipe
   " |"

httpd  root  root

Unix
 Apache mod_logio

mod_log_config

Apache mod_logio
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>%...I</td>
<td>0</td>
</tr>
<tr>
<td>%...O</td>
<td>0</td>
</tr>
</tbody>
</table>

I/O:
"%h %l %u %t "%r" %>s %b "%{Referer}i" " %{User-agent}i" %I %O"
Apache mod_mem_cache

<table>
<thead>
<tr>
<th>URI</th>
<th>Extension</th>
<th>mem_cache_module</th>
<th>mod_mem_cache.c</th>
</tr>
</thead>
</table>

mod_cache mod_cache
mod_proxy ProxyPass( )

mod_cache
don_disk_cache
MCacheMaxObjectCount value
MCacheMaxObjectCount 1009

Extension
mod_mem_cache

MCacheMaxObjectCount
MCacheRemovalAlgorithm

MCacheMaxObjectCount 13001
MCacheMaxObjectSize

MCacheMaxObjectSize 6400000

MCacheMaxObjectSize  10000

Extension
mod_mem_cache

MCacheMaxObjectSize  MCacheMinObjectSize
MCacheMaxStreamingBuffer size_in_bytes
  MCacheMaxStreamingBuffer of 100000
  MCacheMaxObjectSize

Extension
  mod_mem_cache

MCacheMaxStreamingBuffer Length

# Enable caching of streamed responses up to 64KB:
MCacheMaxStreamingBuffer 65536
MCacheMinObjectSize

MCacheMinObjectSize 0

Extension

mod_mem_cache

MCacheMinObjectSize 10000
MCacheRemovalAlgorithm

**LRU (Least Recently Used)**

LRU

**GDSF (GreadyDual-Size)**

GDSF

Extension

mod_mem_cache
<table>
<thead>
<tr>
<th>MCacheSize</th>
<th>KBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCacheSize</td>
<td>100</td>
</tr>
</tbody>
</table>

| Extension | mod_mem_cache |

MCacheSize (1024)

MCacheRemovalAlgorithm

| MCacheSize | 700000 |

| MCacheSize | MCacheMaxObjectSize |

Copyright 2017 The Apache Software Foundation.
Apache mod_mime

AddCharset AddEncoding AddHandler
AddLanguage AddType
content-encoding, content-language, MIME (content-type)
TypesConfig MIME

mod_mime
AddInputFilter
mod_negotiation Multiviews

SetOutputFilter mod_mime

Last-Modified 'touch' ()
MimeMagicFile
AddDefaultCharset
ForceType
DefaultType
SetHandler
SetInputFilter
SetOutputFilter
MIME
UUencoding

HTTP/1.1 RFC 14.11

Content-Encoding
Content-Encoding

(()

Microsoft Word
.zip pkzip

Apache

Content-encoding: pkzip
mime
(MIME
AddType
(MimeMagicFile
AddInputFilter, AddOutputFilter

Charset
Apache

Content-Language: en, fr
Content-Type: text/plain; charset=ISO-8859-1

charset
AddCharset charset extension [extension] ...

, , .htaccess

FileInfo

Base

mod_mime

AddCharset charset

MIME charset

AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .jis


(XXXX.jis.ja) AddCharset charset

extension

• mod_negotiation
• AddDefaultCharset
AddEncoding *MIME-enc* extension [extension]

...,

AddEncoding .htaccess

AddEncoding FileInfo

AddEncoding Base

AddEncoding mod_mime

AddEncoding extension

AddEncoding x-gzip .gz
AddEncoding x-compress .Z

.gz x-gzip

x-zip x-compress

x-Apache compress deflate

extension
AddHandler handler-name extension [extension]  
...  
, .htaccess  
FileInfo  
Base  
mod_mime  

extension handler-name  
".cgi" CGI  

AddHandler cgi-script .cgi  

httpd.conf  
".cgi" CGI  

extension  

- SetHandler
AddInputFilter filter[;filter...] extension [extension] ...

AddInputFilter extension POST

SetInputFilter

- RemoveInputFilter
- SetInputFilter
AddLanguage MIME-lang extension [extension]
  ...
AddLanguage en .en
AddLanguage fr .fr

 xxxx.en.Z compress (language

AddLanguage en .en
AddLanguage en-gb .en
AddLanguage en-us .en

  .en          en-us

  extension

• mod_negotiation
AddOutputFilter  

AddOutputFilter *filter*[, *filter*...]  

extension [extension] ...

AddOutputFilter  

AddOutputFilterByType

```
.shtml SSI  
```

mod_deflate

AddOutputFilter  

Includes;DEFLATE shtml

- RemoveOutputFilter
- SetOutputFilter
AddType MIME-type extension [extension] ...

, , , .htaccess
FileInfo
Base
mod_mime

AddType extension MIME (}

AddType image/gif .gif

MIME TypesConfig Add

extension

- DefaultType
- ForceType
DefaultLanguage Apache

(AddLanguage .fr .de)

AddLanguage en

- mod_negotiation
ModMimeUsePathInfo:

mod_mime:

ModMimeUsePathInfo On

ModMimeUsePathInfo Off

Base:

mod_mime:

Apache 2.0.41

ModMimeUsePathInfo:

mod_mime:

URL:

Off

p

ModMimeUsePathInfo On

/bar (foo.shtml)

ModMimeUsePathInfo Off

/bar/foo.shtml

mod_mime:

AddOutputFilter INCLUDES .shtml

ModMimeUsePathInfo INCLUDES

AcceptPathInfo

• AcceptPathInfo
MultiviewsMatch

Any|NegotiatedOnly|Filters|Handlers
[Handlers|Filters]

MultiviewsMatch NegotiatedOnly

., ., .htaccess

FileInfo

Base

mod_mime

2.0.26

MultiviewsMatch mod_negotiation Multiviews:
Multiviews (index.html)
index.html.fr index.html.gz)

NegotiatedOnly

Multi

500 index.html.cgi 1000 index.html.pl
.cgi .asis .

mod_mime .bak

Multiviews

MultiviewsMatch Handlers Filters

Options

mod_negotiation
RemoveCharset

extension

RemoveCharset .html .shtml
RemoveEncoding

```
/foo/.htaccess:

AddEncoding x-gzip .gz
AddType text/plain .asc
<Files *.gz.asc>
    RemoveEncoding .gz
</Files>
```

foo.gz gzip

RemoveEncoding  AddEncoding

extension
RemoveHandler

/foo/.htaccess:
AddHandler server-parsed .html

/foo/bar/.htaccess:
RemoveHandler .html

/foo/bar .html SSI( mod_include )

delimeter

extension
RemoveInputFilter *extension* `[extension]` ...

., .htaccess

FileInfo

Base

mod_mime

2.0.26

RemoveInputFilter

*extension*

- AddInputFilter
- SetInputFilter
RemoveOutputFilter

extension

RemoveOutputFilter

extension

AddOutputFilter

• AddOutputFilter
<table>
<thead>
<tr>
<th>RemoveType</th>
<th>MIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>/foo/.htaccess:</td>
<td>RemoveType .cgi</td>
</tr>
<tr>
<td>/foo/.cgi</td>
<td>Default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RemoveType</th>
<th>AddType</th>
</tr>
</thead>
<tbody>
<tr>
<td>extension</td>
<td></td>
</tr>
</tbody>
</table>
TypesConfig MIME

IANA

http://www.iana.org/assignments/media-types/index.html

AddType httpd.conf

AddType MIME-type [extension] ...

`('#')

(1) IANA (2) Server Project category/x-subtype

- mod_mime_magic
### Apache Module mod_mime_magic

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Determines the MIME type of a file by looking at a few bytes of its contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module Identifier:</strong></td>
<td>mime_magic_module</td>
</tr>
<tr>
<td><strong>Source File:</strong></td>
<td>mod_mime_magic.c</td>
</tr>
</tbody>
</table>

### Summary

This module determines the [MIME type](#) of files in the same way the Unix `file(1)` command works: it looks at the first few bytes of the file. It is intended as a "second line of defense" for cases that [mod_mime](#) can't resolve.

This module is derived from a free version of the `file(1)` command for Unix, which uses "magic numbers" and other hints from a file's contents to figure out what the contents are. This module is active only if the magic file is specified by the [MimeMagicFile](#) directive.
The contents of the file are plain ASCII text in 4-5 columns. Blank lines are allowed but ignored. Commented lines use a hash mark (#). The remaining lines are parsed for the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>byte number to begin checking from &quot;-&gt;&quot; indicates a dependency upon the previous non-&quot;-&gt;&quot; line</td>
</tr>
<tr>
<td>2</td>
<td>type of data to match</td>
</tr>
<tr>
<td>3</td>
<td>contents of data to match</td>
</tr>
<tr>
<td>4</td>
<td>MIME type if matched</td>
</tr>
<tr>
<td>5</td>
<td>MIME encoding if matched (optional)</td>
</tr>
</tbody>
</table>

For example, the following magic file lines would recognize some audio formats:

`# Sun/NeXT audio data`
Or these would recognize the difference between *.doc files containing Microsoft Word or FrameMaker documents. (These are incompatible file formats which use the same file suffix.)

An optional MIME encoding can be included as a fifth column. For example, this can recognize gzipped files and set the encoding for them.

# gzip (GNU zip, not to be confused with
#     [Info-ZIP/PKWARE] zip archiver)

0 string \037\213 application/octet-stream x-gzip
This module is not for every system. If your system is barely keeping up with its load or if you're performing a web server benchmark, you may not want to enable this because the processing is not free.

However, an effort was made to improve the performance of the original `file(1)` code to make it fit in a busy web server. It was designed for a server where there are thousands of users who publish their own documents. This is probably very common on intranets. Many times, it's helpful if the server can make more intelligent decisions about a file's contents than the file name allows ...even if just to reduce the "why doesn't my page work" calls when users improperly name their own files. You have to decide if the extra work suits your environment.
The following notes apply to the *mod_mime_magic* module and are included here for compliance with contributors' copyright restrictions that require their acknowledgment.

**mod_mime_magic:** MIME type lookup via file magic numbers  
Copyright (c) 1996-1997 Cisco Systems, Inc.

This software was submitted by Cisco Systems to the Apache Group in July 1997. Future revisions and derivatives of this source code must acknowledge Cisco Systems as the original contributor of this module. All other licensing and usage conditions are those of the Apache Group.

Some of this code is derived from the free version of the file command originally posted to comp.sources.unix. Copyright info for that program is included below as required.

- Copyright (c) Ian F. Darwin, 1987. Written by Ian F. Darwin.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must
not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

For compliance with Mr Darwin's terms: this has been very significantly modified from the free "file" command.

- all-in-one file for compilation convenience when moving from one version of Apache to the next.
- Memory allocation is done through the Apache API's pool structure.
- All functions have had necessary Apache API request or server structures passed to them where necessary to call other Apache API routines. (i.e., usually for logging, files, or memory allocation in itself or a called function.)
- struct magic has been converted from an array to a single-ended linked list because it only grows one record at a time, it's only accessed sequentially, and the Apache API has no equivalent of realloc().
- Functions have been changed to get their parameters from the server configuration instead of globals. (It should be reentrant now but has not been tested in a threaded environment.)
- Places where it used to print results to stdout now saves them in a list where they're used to set the MIME type in the Apache request record.
- Command-line flags have been removed since they will never be used here.
**Description:** Enable MIME-type determination based on file contents using the specified magic file

**Syntax:** MimeMagicFile file-path

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_mime_magic

The **MimeMagicFile** directive can be used to enable this module, the default file is distributed at conf/magic. Non-rooted paths are relative to the **ServerRoot**. Virtual hosts will use the same file as the main server unless a more specific setting is used, in which case the more specific setting overrides the main server's file.

**Example**

MimeMagicFile conf/magic
Apache mod_negotiation

- Base
- negotiation_module
- mod_negotiation.c

- (type-map) variants
- MultiViews (MultiViews Option)

Options
mod_mime
**Content-Encoding:**  
Apache AddEncoding compress  
compress gzip x-gzip

**Content-Language:**  
(RFC 1766)

**Content-Length:**  
()

**Content-Type:**  
MIME level  
text/html 2

qs
variant 0.0 1.0  
ASCII ASCII

Content-Type: image/jpeg; qs=0.8

**URI:**  
() variant uri.

**Body:**  
Apache 2.0 Body

**Example:**  

Body:----xyz----
<html>
<body>
<p>Content of the page.</p>
MultiViews  Multiviews Options
/some/dir/foo
Base
mod_negotiation
2.0

HTTP/1.0

CacheNegotiatedDocs on

HTTP/1.1

2.0 CacheNegotiatedDocs on
ForceLanguagePriority

ForceLanguagePriority Prefer HTTP 300 (MULTIPLE CHOICES) LanguagePriority
Accept-Language en de .500 ()

LanguagePriority en fr de
ForceLanguagePriority Prefer

ForceLanguagePriority Fallback HTTP 406 (NOT ACCEPTABLE) LanguagePriority
Language es variant
variant

LanguagePriority en fr de
ForceLanguagePriority Fallback

Prefer Fallback
variant vaient
• AddLanguage
LanguagePriority

| variant |
| LanguagePriority MIME-lang [MIME-lang] ... |
| FileInfo |
| Base |
| mod_negotiation |

**LanguagePriority MultiViews**

**Example:**

LanguagePriority en fr de

foo.html  foo.html.fr  foo.html.de
foo.html.fr

**ForceLang**

- **AddLanguage**
Apache Module mod_nw_ssl

<table>
<thead>
<tr>
<th>Description:</th>
<th>Enable SSL encryption for NetWare</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>Base</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>nwssl_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mod_nw_ssl.c</td>
</tr>
<tr>
<td>Compatibility:</td>
<td>NetWare only</td>
</tr>
</tbody>
</table>

Summary

This module enables SSL encryption for a specified port. It takes advantage of the SSL encryption functionality that is built into the NetWare operating system.
**Description:** List of additional client certificates

**Syntax:**
```
NWSSLTrustedCerts filename [filename]
...
```

**Context:** server config

**Status:** Base

**Module:** mod_nw_ssl

Specifies a list of client certificate files (DER format) that are used when creating a proxied SSL connection. Each client certificate used by a server must be listed separately in its own .der file.
**Description:** Allows a connection to be upgraded to an SSL connection upon request

**Syntax:** NWSSLUpgradeable [IP-address:]portnumber

**Context:** server config

**Status:** Base

**Module:** mod_nw_ssl

Allow a connection that was created on the specified address and/or port to be upgraded to an SSL connection upon request from the client. The address and/or port must have already be defined previously with a `Listen` directive.
**Description:** Enables SSL encryption for the specified port

**Syntax:**

`SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]`

**Context:** server config

**Status:** Base

**Module:** mod_nw_ssl

Specifies the port and the eDirectory based certificate name that will be used to enable SSL encryption. An optional third parameter also enables mutual authentication.
Apache mod_proxy

ProxyRequests

Apache / AJP13 (Apache JServe Protocol version 1.3),
CONNECT (SSL), HTTP/0.9, HTTP/1.0, HTTP/1.1

Apache mod_proxy : mod_proxy_ftp, mod_proxy_ajp, mod_proxy_balancer,
mod_proxy_connect)

mod_cache
mod_proxy_http
mod_proxy_ftp
mod_proxy_connect
mod_proxy_balancer
mod_ssl
Apache

ProxyRequests

ProxyPass

RewriteRule [P]
mod_cache

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

ProxyRequests On
ProxyVia On

<Proxy *
    Order deny,allow
    Deny from all
    Allow from internal.example.com
</Proxy>
ProxyPass /example http://backend.example.com
connectiontimeout=5 timeout=30

http://backend.example.com URL
ProxySet:

ProxySet http://backend.example.com connectiontimeout=5 timeout=30

Proxy  ProxySet:

<Proxy http://backend.example.com>
    ProxySet connectiontimeout=5 timeout=30
</Proxy>

ProxyPass /examples http://backend.example.com/examples
ProxyPass /docs http://backend.example.com/docs
ProxyPass /apps http://backend.example.com/ timeout=60
ProxyPass /examples http://backend.example.com/examples timeout=10
<Proxy>  

<Proxy *
 Order Deny,Allow
 Deny from all
 Allow from 192.168.0
</Proxy>

mod_authz_host

(ProxyRequests )
(ProxyRequests Off ProxyPass )
ProxyBlock
Apache

WWW
"http://somehost/"

Apache

NoProxy

(full qualify)
Keepalive  HTTP/1.1
keepalive

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
mod_proxy_http Content-Length
chunked Length proxy-sendchunked
(ProxyPass) mod_proxy_http

X-Forwarded-For
  IP

X-Forwarded-Host
  Host

X-Forwarded-Server

ProxyPreserveHost  ProxyVia
CONNECT
AllowCONNECT port [port] ...
AllowCONNECT 443 563

Extension
mod_proxy

AllowCONNECT CONNECT

https (443) snews (563)
AllowCONNECT

CONNECT mod_proxy_connect
BalancerMember [balancerurl] url [key=value [key=value ...]]

Extension
mod_proxy
BalancerMember Apache 2.2

<Proxy balancer://...
ProxyPass URL
NoProxy host [host] ...

Extension
mod_proxy

Apache

ProxyRemote

ProxyRemote * http://firewall.example.com:81
NoProxy .example.com 192.168.112.0/21

NoProxy host :

Domain

Domain DNS


Domain Hostname (DNS !) Domain

Domain DNS

.example.com. () DNS

SubNet

SubNet ()

(8) :
<table>
<thead>
<tr>
<th>SubNet</th>
<th>IPAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0/16</td>
<td>192.168.0.0/16</td>
</tr>
<tr>
<td>192.168.112.0/21</td>
<td>192.168.112.0/21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SubNet</th>
<th>IPAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0/16</td>
<td>(255.255.0.0)</td>
</tr>
<tr>
<td>32</td>
<td>SubNet <em>Default</em></td>
</tr>
</tbody>
</table>

**IPAddr**

<table>
<thead>
<tr>
<th>IPAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IPAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.123.7</td>
</tr>
</tbody>
</table>

**Hostname**

<table>
<thead>
<tr>
<th>Hostname</th>
<th>DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>()</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>prep.ai.example.com</td>
</tr>
<tr>
<td><a href="http://www.apache.org">www.apache.org</a></td>
</tr>
</tbody>
</table>

**Hostname**

<table>
<thead>
<tr>
<th>Hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.example.com.()">www.example.com.()</a></td>
</tr>
</tbody>
</table>
• DNS
<Proxy wildcard-url> ...</Proxy>

Extension

mod_proxy

<Proxy>
  yournetwork.example.com :
</Proxy>

<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from yournetwork.example.com
</Proxy>

example.com foo

<Proxy http://example.com/foo/*>
  SetOutputFilter INCLUDES
</Proxy>

- <ProxyMatch>
ProxyBadHeader

IsError

502 (Bad Gateway)

Ignore

StartBody
ProxyBlock

Example

ProxyBlock joes-garage.com some-host.co.uk rocky.wotsamattau.edu

rocky.wotsamattau.edu IP

wotsamattau.edu wotsamattau

ProxyBlock *
Apache

ProxyRemote * http://firewall.example.com:81
NoProxy .example.com 192.168.112.0/21
ProxyDomain .example.com
ProxyErrorOverride On|Off
ProxyErrorOverride Off

Extension
mod_proxy
Apache 2.0

informational (1xx), (2xx), (3xx)
ProxyFtpDirCharset character set
ProxyFtpDirCharset ISO-8859-1
Extension
mod_proxy
Apache 2.2.7
<table>
<thead>
<tr>
<th>ProxyIOBufferSize</th>
<th>bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProxyIOBufferSize</td>
<td>8192</td>
</tr>
</tbody>
</table>

Extension:
- `mod_proxy`
<ProxyMatch>
  
  <ProxyMatch regex> ...</ProxyMatch>

  
  Extension
  
  mod_proxy

<ProxyMatch> URL <Proxy>

- <Proxy>

<Proxy>
ProxyMaxForwards:

ProxyMaxForwards number

ProxyMaxForwards -1

Extension

mod_proxy

Apache 2.0 ; Apache 2.2.7

ProxyMaxForwards Max-Forwards

ProxyMaxForwards 15

ProxyMaxForwards HTTP/1.1 (RFC2616) Max-Forwards

RFC2616

HTTP/1.1
ProxyPass
ProxyRequests

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/i  /mirror/foo
backend.example.com

ProxyPass
ProxyPassMatch
URL
URL
ProxyPass

Apache HTTP 2.1

mod_proxy Web

ProxyPass /example http://backend.example.com max=20 ttl=120 retry=300

min 0
max 1...n
smax max ttl
acquire -
connectiontimeout timeout
disablereuse Off
flushpackets off

MPM 1 MPM
ThreadsPerChild
SERVER_BUSY
AJP
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>flushwait</td>
<td>10</td>
<td><code>'flushpackets' 'auto'</code></td>
</tr>
<tr>
<td>keepalive</td>
<td>Off</td>
<td>Apache OS</td>
</tr>
<tr>
<td>TCP OS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lbset</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ping</td>
<td>0</td>
<td>Web ajp13</td>
</tr>
<tr>
<td>loadfactor</td>
<td>1</td>
<td>BalancerMember 100</td>
</tr>
<tr>
<td>redirect</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>retry</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>route</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>-</td>
<td>: 'D' (disabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(stopped) 'I' (ignore-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>errors) 'H' (hot-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>standby) 'E' (error)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'S-E'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td><code>ProxyTimeout</code></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ProxyPass</td>
<td><code>balancer:// (balancer://cluster/) mod_proxy_balancer</code></td>
<td></td>
</tr>
</tbody>
</table>

**lbmethod**
- `byrequests`  
  - Balancer byrequests
  - Balancer (Apache HTTP 2.2.10)
  - `byrequests`

**maxattempts**
- 1

**nofailover**
- Off
  - `On`

**stickysession**
- `- JSESSIONID PHPSESSIONI`

**scolonpathdelim**
- Off
  - `On ';'`

**URL**
- `JSESSIONID=6736bcf34;foo=aa`

**timeout**
- 0

**failonstatus**
- `- HTTP Apache HTTP 2.2.17`

---

ProxyPass /special-area http://special.example.com smax=5 max=10
ProxyPass / balancer://mycluster/
stickysession=JSESSIONID|jsessionid nofailover=On

<Proxy balancer://mycluster>
  
  - `BalancerMember ajp://1.2.3.4:8009`
  - `BalancerMember ajp://1.2.3.5:8009 loadfactor=20`
  - `# Less powerful server, don't send as many requests there,`
  - `BalancerMember ajp://1.2.3.6:8009 loadfactor=5`
ProxyPass / balancer://hotcluster/
<Proxy balancer://hotcluster>
  BalancerMember ajp://1.2.3.4:8009 loadfactor=1
  BalancerMember ajp://1.2.3.5:8009 loadfactor=2
  # The below is the hot standby
  BalancerMember ajp://1.2.3.6:8009 status=+H
  ProxySet lbmethod=bytraffic
</Proxy>

mod_proxy ProxyPass URL
  PATH_INFO nocanon URL ("raw") URL

  interpolate (httpd 2.2.9)
ProxyPassInterpolateEnv ${VARNAME}
ProxyPass CGI

<Location>
  <LocationMatch> ProxyPass

  <Directory> <Files>

    [P] RewriteRule

  </Files>

</Directory>
ProxyPassInterpolateEnv On|Off
ProxyPassInterpolateEnv Off

Extension
mod_proxy
Apache 2.2.9

ProxyPass, ProxyPassReverse,
ProxyPassReverseCookieDomain,
ProxyPassReverseCookiePath interpolate
mod_rewrite
ProxyPass
ProxyPassReverseCookieDomain,
ProxyPassReverseCookiePath
varname (interpolate)
()
ProxyPassMatch [regex] !|url [key=value [key=value ...]]

Extension
mod_proxy
Apache 2.2.5

http://example.com/;

ProxyPassMatch ^(/.*\.gif)$ http://backend.example.com$1

http://example.com/foo/bar.gif
http://backend.example.com/foo/bar.gif

ProxyPassMatch ^(.*\.gif)$ http://backend.example.com:8000$1

!(ASF bugzilla PR 46665)

ProxyPassMatch ^/(.*\.gif)$ http://backend.example.com:8000/$1

<LocationMatch>
ProxyPassReverse [path] url [interpolate]

Extension
mod_proxy

Apache HTTP
Location, Content-Location
Apache (
HTTP
URL

path url URL

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
) backend.example.com
    http://backend.example.com/bar
http://backend.example.com/quux

UseCanonicalName

ProxyPassReverse
(RewriteRule ... [P])
interpolate (httpd 2.2.9)
ProxyPassInterpolateEnv ${VARNAME}
ProxyPassReverse

<Location>
<LocationMatch> ProxyPassReverse
ProxyPassReverse
</Location>

<Directory> <Files>
Set-Cookie Domain

ProxyPassReverseCookieDomain internal-domain
public-domain [interpolate]

Extension

mod_proxy

ProxyPassReverse URL Set-Cookie domain
<table>
<thead>
<tr>
<th>Set-Cookie Path</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ProxyPassReverseCookiePath</strong></td>
<td><strong>internal-path</strong></td>
</tr>
<tr>
<td><strong>public-path</strong></td>
<td><strong>[interpolate]</strong></td>
</tr>
</tbody>
</table>

| Extension | mod_proxy |

**URL**

| path | **internal-path** | **public-path** |

**ProxyPassReverse**:

```
ProxyPassReverseCookiePath / /mirror/foo/
```

/( /example) /mirror/foo/
ProxyPreserveHost Off

Extension mod_proxy

Apache 2.0.31

ProxyPass Host: Off
ProxyReceiveBufferSize

HTTP	FTP

ProxyReceiveBufferSize: bytes
ProxyReceiveBufferSize: 0

Extension: mod_proxy

ProxyReceiveBufferSize (TCP/IP)

ProxyReceiveBufferSize: 2048
ProxyRemote match remote-server

Extension mod_proxy

match \n
URL:

remote-server = scheme://hostname[(:port)]

scheme

ProxyRemote http://goodguys.example.com/
http://mirrorguys.example.com:8000
ProxyRemote * http://cleverproxy.localdomain

FTP HTTP
ProxyRemoteMatch

ProxyRemoteMatch regex remote-server

Extension

mod_proxy
Apache

/  Off

HTTP FTP

mod_proxy

ProxyRequests
ProxySet url key=value [key=value ...

Extension
mod_proxy
ProxySet Apache 2.2

ProxyPass
url

<Proxy balancer://hotcluster>
  BalancerMember http://www2.example.com:8009 loadfactor=1
  BalancerMember http://www3.example.com:8009 loadfactor=2
  ProxySet lbmethod=bytraffic
</Proxy>

<Proxy http://backend>
  ProxySet keepalive=On
</Proxy>

ProxySet balancer://foo lbmethod=bytraffic timeout=15

ProxySet ajp://backend:7001 timeout=15
ProxyStatus:

mod_status
ProxyStatus Off|On|Full
ProxyStatus Off,

Extension
mod_proxy
Apache 2.2

mod_status

Full On
ProxyTimeout *seconds*

**Timeout**

Extension

*mod_proxy*

Apache 2.0.31
Via HTTP

ProxyVia On|Off|Full|Block

ProxyVia Off

,   

Extension

mod_proxy
Apache mod_proxy_ajp

mod_proxy AJP
Extension
proxy_ajp_module
mod_proxy_ajp.c

mod_proxy Apache JServ Protocol version 1.3 (AJP13)

mod_proxy
AJP13

- Idle
- Assigned

(HTTP > 0)

- SEND_HEADERS
- SEND_BODY_CHUNK
- GET_BODY_CHUNK
- END_RESPONSE
XDR

4 : byte, boolean, integer, string

Byte

Boolean

1 = true, 0 = false

Integer

0 \ 2^{16} (32768) \ 2

String

(2^{16}) 2

\text{strlen}
\ \ \ \ 0 \ C

8 * 1024

0x1234

AB (ASCII \ /)

) \ 2^{16}

(Server->Container)

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4...(n+3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>0x12</td>
<td>0x34</td>
<td>(n)</td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

(Container->Server)

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4...(n+3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>A</td>
<td>B</td>
<td>(n)</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Forward Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Shutdown</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Ping</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CPing, CPong</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data | Forward Request |
:---|:---|

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Send Body Chunk</td>
</tr>
<tr>
<td>4</td>
<td>Send Headers</td>
</tr>
<tr>
<td>5</td>
<td>End Response</td>
</tr>
<tr>
<td>6</td>
<td>Get Body Chunk</td>
</tr>
<tr>
<td>9</td>
<td>CPong, CPing</td>
</tr>
</tbody>
</table>
Forward Request:

AJP13_FORWARD_REQUEST :=
  prefix_code (byte) 0x02 = JK_AJP13_FORWARD_REQUEST
  method (byte)
  protocol (string)
  req_uri (string)
  remote_addr (string)
  remote_host (string)
  server_name (string)
  server_port (integer)
  is_ssl (boolean)
  num_headers (integer)
  request_headers *(req_header_name req_header_value)
  attributes *(attribut_name attribute_value)
  request_terminator (byte) 0xFF

request_headers:

req_header_name :=
  sc_req_header_name | (string) [see below for how this is parsed]

sc_req_header_name := 0xA0xx (integer)

req_header_value := (string)

:

attribute_name := sc_a_name | (sc_a_req_attribute string)

attribute_value := (string)

content-length

Forward Request

Request prefix

2 Prefix
### Method

**HTTP 1:**

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIONS</td>
<td>1</td>
</tr>
<tr>
<td>GET</td>
<td>2</td>
</tr>
<tr>
<td>HEAD</td>
<td>3</td>
</tr>
<tr>
<td>POST</td>
<td>4</td>
</tr>
<tr>
<td>PUT</td>
<td>5</td>
</tr>
<tr>
<td>DELETE</td>
<td>6</td>
</tr>
<tr>
<td>TRACE</td>
<td>7</td>
</tr>
<tr>
<td>PROPFIND</td>
<td>8</td>
</tr>
<tr>
<td>PROPPATCH</td>
<td>9</td>
</tr>
<tr>
<td>MKCOL</td>
<td>10</td>
</tr>
<tr>
<td>COPY</td>
<td>11</td>
</tr>
<tr>
<td>MOVE</td>
<td>12</td>
</tr>
<tr>
<td>LOCK</td>
<td>13</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>14</td>
</tr>
<tr>
<td>ACL</td>
<td>15</td>
</tr>
<tr>
<td>REPORT</td>
<td>16</td>
</tr>
<tr>
<td>VERSION-CONTROL</td>
<td>17</td>
</tr>
<tr>
<td>CHECKIN</td>
<td>18</td>
</tr>
<tr>
<td>CHECKOUT</td>
<td>19</td>
</tr>
<tr>
<td>UNCHECKOUT</td>
<td>20</td>
</tr>
<tr>
<td>SEARCH</td>
<td>21</td>
</tr>
<tr>
<td>MKWORKSPACE</td>
<td>22</td>
</tr>
<tr>
<td>UPDATE</td>
<td>23</td>
</tr>
<tr>
<td>LABEL</td>
<td>24</td>
</tr>
<tr>
<td>MERGE</td>
<td>25</td>
</tr>
<tr>
<td>BASELINE_CONTROL</td>
<td>26</td>
</tr>
</tbody>
</table>
ajp13

**protocol, req_uri, remote_addr, remote_host, server_name, server_port, is_ssl**

**Headers**

```java
Headers
request_headers: num_headers
req_header_name/req_header_value
)
```

<table>
<thead>
<tr>
<th>Header</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>0xA001 SC_REQ_ACCEPT</td>
</tr>
<tr>
<td>accept-charset</td>
<td>0xA002 SC_REQ_ACCEPT_CHARSET</td>
</tr>
<tr>
<td>accept-encoding</td>
<td>0xA003 SC_REQ_ACCEPT_ENCODING</td>
</tr>
<tr>
<td>accept-language</td>
<td>0xA004 SC_REQ_ACCEPT_LANGUAGE</td>
</tr>
<tr>
<td>authorization</td>
<td>0xA005 SC_REQ_AUTHORIZATION</td>
</tr>
<tr>
<td>connection</td>
<td>0xA006 SC_REQ_CONNECTION</td>
</tr>
<tr>
<td>content-type</td>
<td>0xA007 SC_REQ_CONTENT_TYPE</td>
</tr>
<tr>
<td>content-length</td>
<td>0xA008 SC_REQ_CONTENT_LENGTH</td>
</tr>
<tr>
<td>cookie</td>
<td>0xA009 SC_REQ_COOKIE</td>
</tr>
<tr>
<td>cookie2</td>
<td>0xA00A SC_REQ_COOKIE2</td>
</tr>
<tr>
<td>host</td>
<td>0xA00B SC_REQ_HOST</td>
</tr>
<tr>
<td>pragma</td>
<td>0xA00C SC_REQ_PRAGMA</td>
</tr>
<tr>
<td>referer</td>
<td>0xA00D SC_REQ_REFERER</td>
</tr>
<tr>
<td>user-agent</td>
<td>0xA00E SC_REQ_USER_AGENT</td>
</tr>
</tbody>
</table>
```

Java 2
0x9999 (==0xA000 -1)

:
content-length ( POST

? ( ?context) 1

<table>
<thead>
<tr>
<th>Information</th>
<th>Code</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>?context</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>?servlet_path</td>
<td>0x02</td>
<td></td>
</tr>
<tr>
<td>?remote_user</td>
<td>0x03</td>
<td></td>
</tr>
<tr>
<td>?auth_type</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>?query_string</td>
<td>0x05</td>
<td></td>
</tr>
<tr>
<td>?jvm_route</td>
<td>0x06</td>
<td></td>
</tr>
<tr>
<td>?ssl_cert</td>
<td>0x07</td>
<td></td>
</tr>
<tr>
<td>?ssl_cipher</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>?ssl_session</td>
<td>0x09</td>
<td></td>
</tr>
<tr>
<td>?req_attribute</td>
<td>0x0A</td>
<td>Name (the name of the attribute follows)</td>
</tr>
<tr>
<td>?ssl_key_size</td>
<td>0x0B</td>
<td></td>
</tr>
<tr>
<td>are_done</td>
<td>0xFF</td>
<td>request_terminator</td>
</tr>
</tbody>
</table>

context servlet_path C Java )
remote_user auth_type HTTP
query_string, ssl_cert, ssl_cipher, ssl_session
HTTP HTTPS

jvm_route —— Tomcat

req_attribute 0x0A

0xFF
Send Body Chunk

Send Headers

HTTP (200 OK)
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

End Response
   reuse  (==1) TCP

Get Body Chunk

(          )
- 6)
(          )
3
Counting) (Pending Request Counting)
ProxyPass
| mod_proxy_balancer | 2 URL URL |
<Proxy balancer://mycluster>
BalancerMember http://192.168.1.50:80
BalancerMember http://192.168.1.51:80
</Proxy>
ProxyPass /test balancer://mycluster

mod_headers

Header add Set-Cookie "ROUTEID=%.{BALANCER_WORKER_ROUTE}e;
path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://mycluster>
BalancerMember http://192.168.1.50:80 route=1
BalancerMember http://192.168.1.51:80 route=2
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /test balancer://mycluster
lbmethod=byrequests

lbfactor

lbstatus

for each worker in workers
    worker lbstatus += worker lbfactor
    total factor += worker lbfactor
    if worker lbstatus > candidate lbstatus
        candidate = worker

candidate lbstatus -= total factor

: 

<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbfactor</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>lbstatus</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

b

<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbstatus</td>
<td>-50</td>
<td>0</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>lbstatus</td>
<td>-25</td>
<td>0</td>
<td>-25</td>
<td>50</td>
</tr>
<tr>
<td>lbstatus</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(repeat)

: a c d a c d a c d ... :
<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbfactor</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

: 

<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbfactor</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

lbfactor

: 

<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbfactor</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

b a c 4

: 

<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbfactor</td>
<td>70</td>
<td>30</td>
</tr>
</tbody>
</table>

| 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)

10 a 7 b 3
Weighted Traffic Counting

lbmethod=bytraffic

Request Counting

lbfactor

<table>
<thead>
<tr>
<th>worker</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbfactor</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

b a c 2

b 2 I/O
lbmethod=bybusyness

Request Counting

Apache HTTP 2.2.10
6:

**BALANCER_SESSION_STICKY**

stickysession

**BALANCER_SESSION_ROUTE**

route

**BALANCER_NAME**

balancer://foo

**BALANCER_WORKER_NAME**

http://hostA:1234

**BALANCER_WORKER_ROUTE**

route

**BALANCER_ROUTE_CHANGED**

(BALANCER_SESSION_ROUTE != BALANCER_WORKER_ROUTE) 1
mod_status

mod_status  mod_prox

e.example.com

<Location /balancer-manager>
SetHandler balancer-manager

Order Deny,Allow
Deny from all
Allow from .example.com
</Location>

http://your.server.name/balancer-manager
route
Apache Web

route ProxyPass
Apache Web

conf/server.xml jvmRoute Tomcat
Tomcat Java Web

URL
Web URL

Java URL
URL (
jvmRoute Apache
scolonpathdelim On

URL

ProxyPass /test balancer://mycluster
stickySession=JSESSIONID|jsessionid scolonpathdelim=On
<Proxy balancer://mycluster>
BalancerMember http://192.168.1.50:80 route=node1
BalancerMember http://192.168.1.51:80 route=node2
</Proxy>
URL

```
%{MYCOOKIE}C
MYCOOKIE

%{Set-Cookie}o

%{BALANCER_SESSION_STICKY}e

%{BALANCER_SESSION_ROUTE}e
  route

%{BALANCER_WORKER_ROUTE}e
  route

%{BALANCER_ROUTE_CHANGED}e
  route  route
  1

debug
```
Apache mod_proxy_connect

CONNECT

mod_proxy

Extension

proxy_connect_module

mod_proxy_connect.c

mod_proxy CONNECT HTTP SSL

CONNECT

mod_proxy mod_proxy_conn

AllowCONNECT

mod_proxy

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0
Apache mod_proxy_ftp

mod_proxy FTP

Extension
proxy_ftp_module
mod_proxy_ftp.c

mod_proxy FTP

mod_proxy mod_proxy_ftp

mod_proxy
<table>
<thead>
<tr>
<th>mime.types</th>
<th>application/octet-stream</th>
<th>bin</th>
<th>dms</th>
<th>lha</th>
<th>lzh</th>
<th>exe</th>
<th>class</th>
<th>tgz</th>
<th>taz</th>
</tr>
</thead>
</table>

: 

DefaultType application/octet-stream
FTP    ASCII
mod_proxy ASCII

(FTP)
mod_proxy  FTP  GET
(POST  PUT)
| FTP URI |  
|---------|---
| FTP URI | /../
| FTP URI | Squid %2f hack ;
| FTP URI | Squid Proxy Cache
| FTP URI | / ()

ftp://user@host/%2f/etc/motd
### FTP

**Apache URL**

<table>
<thead>
<tr>
<th>user: anonymous</th>
<th>password: apache_proxy@</th>
</tr>
</thead>
</table>

**FTP URL**

```
ftp://username@host/myfile
```

**FTP ()**

```
ftp://username:password@host/myfile
```

**HTTP FTP (FTP)**

```
Apache basic
```

---

Copyright 2017 The Apache Software Foundation.  
[Apache License, Version 2.0](https://www.apache.org/licenses/LICENSE-2.0).
Apache mod_proxy_http

mod_proxy HTTP
Extension
proxy_http_module
mod_proxy_http.c

mod_proxy HTTP HTTPS
HTTP/0.9 HTTP/1.0 HTTP/1.1

HTTP mod_proxy mod_proxy_http

mod_proxy
mod_proxy_connect
mod_proxy HTTP

proxy-sendextracrlf
   CR-LF

force-proxy-request-1.0
   HTTP/1.0 HTTP/1.1

proxy-nokeepalive

proxy-chain-auth

proxy-sendcl
   HTTP/1.0 (POST)
   Apache

proxy-sendchunks or proxy-sendchunked
   proxy-sendcl chunked

proxy-interim-response
   RFC Suppress httpd HTTP interim (1xx)
   HTTP interim
   proxy-interim-response RFC
   interim-response Suppress interim

proxy-initial-not-pooled
Apache mod_proxy_scgi

- mod_proxy
- SCGI
- Extension
- proxy_scgi_module
- mod_proxy_scgi.c
- Apache 2.2.14

mod_proxy SCGI protocol, version 1

SCGI mod_proxy mod_proxy_scgi

mod_proxy mod_proxy_balancer
mod_proxy  mod_proxy_scgi

ProxyPass /scgi-bin/ scgi://localhost:4000/

mod

ProxyPass /scgi-bin/ balancer://somecluster/
<Proxy balancer://somecluster/>  
  BalancerMember scgi://localhost:4000/  
  BalancerMember scgi://localhost:4001/  
</Proxy>
<table>
<thead>
<tr>
<th>ProxySCGIInternalRedirect</th>
<th>URL</th>
</tr>
</thead>
</table>
| On                       | mod_proxy_scgi 0K (200) | mod_cgi | Location ( )

ProxySCGIInternalRedirect Off
<table>
<thead>
<tr>
<th>X-Sendfile</th>
<th>ProxySCGISendfile On</th>
<th>Off</th>
<th>Headername</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ProxySCGISendfile Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Extension</td>
</tr>
<tr>
<td></td>
<td>mod_proxy_scgi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```yaml
ProxySCGISendfile: SCGI

ProxySCGISendfile: Off

On
  X-Sendfile
  Sendfile.

On

Example
# Use the default header (X-Sendfile)
ProxySCGISendfile On

# Use a different header
ProxySCGISendfile X-Send-Static
```
## Apache Module mod_reqtimeout

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Set timeout and minimum data rate for receiving requests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module Identifier:</strong></td>
<td>reqtimeout_module</td>
</tr>
<tr>
<td><strong>Source File:</strong></td>
<td>mod_reqtimeout.c</td>
</tr>
<tr>
<td><strong>Compatibility:</strong></td>
<td>Available in Apache 2.2.15 and later</td>
</tr>
</tbody>
</table>
1. Allow 10 seconds to receive the request including the headers and 30 seconds for receiving the request body:

```
RequestReadTimeout header=10 body=30
```

2. Allow at least 10 seconds to receive the request body. If the client sends data, increase the timeout by 1 second for every 1000 bytes received, with no upper limit for the timeout (except for the limit given indirectly by `LimitRequestBody`):

```
RequestReadTimeout body=10,MinRate=1000
```

3. Allow at least 10 seconds to receive the request including the headers. If the client sends data, increase the timeout by 1 second for every 500 bytes received. But do not allow more than 30 seconds for the request including the headers:

```
RequestReadTimeout header=10-30,MinRate=500
```

4. Usually, a server should have both header and body timeouts configured. If a common configuration is used for http and https virtual hosts, the timeouts should not be set too low:

```
RequestReadTimeout header=20-40,MinRate=500
body=20,MinRate=500
```
**Description:** Set timeout values for receiving request headers and body from client.

**Syntax:**
```
RequestReadTimeout [header=timeout[[maxtimeout],MinRate=rate] [body=timeout[maxtimeout],MinRate=rate]
```

**Default:** Unset; no limit

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_reqtimeout

This directive can set various timeouts for receiving the request headers and the request body from the client. If the client fails to send headers or body within the configured time, a 408 REQUEST TIME OUT error is sent.

For SSL virtual hosts, the header timeout values include the time needed to do the initial SSL handshake. If the user's browser is configured to query certificate revocation lists and the CRL server is not reachable, the initial SSL handshake may take a significant time until the browser gives up waiting for the CRL. Therefore the header timeout values should not be set to very low values for SSL virtual hosts. The body timeout values include the time needed for SSL renegotiation (if necessary).

When an **AcceptFilter** is in use (usually the case on Linux and FreeBSD), the socket is not sent to the server process before at least one byte (or the whole request for httpready) is received. The header timeout configured with RequestReadTimeout is only effective after the server process has received the socket.

For each of the two timeout types (header or body), there are three ways to specify the timeout:
**Fixed timeout value:**

```plaintext
type=timeout
```

The time in seconds allowed for reading all of the request headers or body, respectively. A value of 0 means no limit.

**Timeout value that is increased when data is received:**

```plaintext
type=timeout, MinRate=data_rate
```

Same as above, but whenever data is received, the timeout value is increased according to the specified minimum data rate (in bytes per second).

**Timeout value that is increased when data is received, with an upper bound:**

```plaintext
type=timeout-maxtimeout, MinRate=data_rate
```

Same as above, but the timeout will not be increased above the second value of the specified timeout range.
Apache Module mod_rewrite

<table>
<thead>
<tr>
<th>Description:</th>
<th>Provides a rule-based rewriting engine to rewrite requested URLs on the fly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status:</td>
<td>Extension</td>
</tr>
<tr>
<td>Module Identifier:</td>
<td>rewrite_module</td>
</tr>
<tr>
<td>Source File:</td>
<td>mod_rewrite.c</td>
</tr>
<tr>
<td>Compatibility:</td>
<td>Available in Apache 1.3 and later</td>
</tr>
</tbody>
</table>

Summary

This module uses a rule-based rewriting engine (based on a regular-expression parser) to rewrite requested URLs on the fly. It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule, to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests, of server variables, environment variables, HTTP headers, or time stamps. Even external database lookups in various formats can be used to achieve highly granular URL matching.

This module operates on the full URLs (including the path-info part) both in per-server context (httpd.conf) and per-directory context (.htaccess) and can generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to an internal proxy throughput.

Further details, discussion, and examples, are provided in the detailed mod_rewrite documentation.

See also

Rewrite Flags
As of Apache 1.3.20, special characters in **TestString** and **Substitution** strings can be escaped (that is, treated as normal characters without their usual special meaning) by prefixing them with a backslash ('\') character. In other words, you can include an actual dollar-sign character in a **Substitution** string by using '\$'; this keeps mod_rewrite from trying to treat it as a backreference.
This module keeps track of two additional (non-standard) CGI/SSI environment variables named SCRIPT_URL and SCRIPT_URI. These contain the logical Web-view to the current resource, while the standard CGI/SSI variables SCRIPT_NAME and SCRIPT_FILENAME contain the physical System-view.

Notice: These variables hold the URI/URL as they were initially requested, that is, before any rewriting. This is important to note because the rewriting process is primarily used to rewrite logical URLs to physical pathnames.

**Example**

SCRIPT_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.html
SCRIPT_FILENAME=/u/rse/.www/index.html
SCRIPT_URL=/u/rse/
SCRIPT_URI=http://en1.engelschall.com/u/rse/
By default, *mod_rewrite* configuration settings from the main server context are not inherited by virtual hosts. To make the main server settings apply to virtual hosts, you must place the following directives in each `<VirtualHost>` section:

```
RewriteEngine On
RewriteOptions Inherit
```
For numerous examples of common, and not-so-common, uses for mod_rewrite, see the extended rewrite documentation.
The **RewriteBase** directive specifies the URL prefix to be used for per-directory (htaccess) **RewriteRule** directives that substitute a relative path.

This directive is *required* when you use a relative path in a substitution in per-directory (htaccess) context unless either of the following conditions are true:

- The original request, and the substitution, are underneath the [DocumentRoot](#) (as opposed to reachable by other means, such as [Alias](#)).
- The *filesystem* path to the directory containing the [RewriteRule](#), suffixed by the relative substitution is also valid as a URL path on the server (this is rare).

In the example below, **RewriteBase** is necessary to avoid rewriting to `http://example.com/opt/myapp-1.2.3/welcome.html` since the resource was not relative to the document root. This misconfiguration would normally cause the server to look for an "opt" directory under the document root.
RewriteRule ^index\.html$ welcome.html
</Directory>
The `RewriteCond` directive defines a rule condition. One or more `RewriteCond` can precede a `RewriteRule` directive. The following rule is then only used if both the current state of the URI matches its pattern, and if these conditions are met.

`TestString` is a string which can contain the following expanded constructs in addition to plain text:

- **RewriteRule backreferences**: These are backreferences of the form $N (0 <= N <= 9), which provide access to the grouped parts (in parentheses) of the pattern, from the RewriteRule which is subject to the current set of RewriteCond conditions.
- **RewriteCond backreferences**: These are backreferences of the form %N (1 <= N <= 9), which provide access to the grouped parts (again, in parentheses) of the pattern, from the last matched RewriteCond in the current set of conditions.
- **RewriteMap expansions**: These are expansions of the form $\{mapname: key|default\}. See the documentation for RewriteMap for more details.
- **Server-Variables**: These are variables of the form %\{ NAME_OF_VARIABLE \} where NAME_OF_VARIABLE can be a string taken from the following list:

<table>
<thead>
<tr>
<th>HTTP headers:</th>
<th>connection &amp;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Request Variables</td>
<td>Server Internals</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>HTTP_USER_AGENT, HTTP_REFERER, HTTP_COOKIE, HTTP_FORWARDED, HTTP_HOST, HTTP_PROXY_CONNECTION, HTTP_ACCEPT</td>
<td>DOCUMENT_ROOT, SERVER_ADMIN, SERVER_NAME, SERVER_ADDR, SERVER_PORT, SERVER_PROTOCOL, SERVER_SOFTWARE</td>
</tr>
</tbody>
</table>

These variables all correspond to the similarly named HTTP MIME-headers, C variables of the Apache server or struct tm fields of the Unix system. Most are documented elsewhere in the Manual or in the CGI specification.

SERVER_NAME and SERVER_PORT depend on the values of UseCanonicalName and UseCanonicalPhysicalPort respectively.

Those that are special to mod_rewrite include those below.

IS_SUBREQ
Will contain the text "true" if the request currently being processed is a sub-request, "false" otherwise. Sub-requests may be generated by modules that need to resolve additional files or URIs in order to complete their tasks.

**API_VERSION**
This is the version of the Apache module API (the internal interface between server and module) in the current httpd build, as defined in include/ap_mmn.h. The module API version corresponds to the version of Apache in use (in the release version of Apache 1.3.14, for instance, it is 19990320:10), but is mainly of interest to module authors.

**THE_REQUEST**
The full HTTP request line sent by the browser to the server (e.g., "GET /index.html HTTP/1.1"). This does not include any additional headers sent by the browser. This value has not been unescaped (decoded), unlike most other variables below.

**REQUEST_URI**
The path component of the requested URI, such as "/index.html". This notably excludes the query string which is available as its own variable named QUERY_STRING.

**REQUEST_FILENAME**
The full local filesystem path to the file or script matching the request, if this has already been determined by the server at the time REQUEST_FILENAME is referenced. Otherwise, such as when used in virtual host context, the same value as REQUEST_URI.

**HTTPS**
Will contain the text "on" if the connection is using SSL/TLS, or "off" otherwise. (This variable can be safely used regardless of whether or not `mod_ssl` is loaded).

Other things you should be aware of:

1. The variables `SCRIPT_FILENAME` and `REQUEST_FILENAME` contain the same value - the value of the filename field of the internal `request_rec` structure of the Apache server. The first name is the commonly known CGI variable name while the second is the appropriate counterpart of `REQUEST_URI` (which contains the value of the `uri` field of `request_rec`).

   If a substitution occurred and the rewriting continues, the value of both variables will be updated accordingly.

   If used in per-server context (i.e., before the request is mapped to the filesystem) `SCRIPT_FILENAME` and `REQUEST_FILENAME` cannot contain the full local filesystem path since the path is unknown at this stage of processing. Both variables will initially contain the value of `REQUEST_URI` in that case. In order to obtain the full local filesystem path of the request in per-server context, use an URL-based look-ahead `%{LA-U:REQUEST_FILENAME}` to determine the final value of `REQUEST_FILENAME`.

2. `%{ENV:variable}`, where `variable` can be any environment variable, is also available. This is looked-up via internal Apache structures and (if not found there) via `getenv()` from the Apache server process.

3. `%{SSL:variable}`, where `variable` is the name of an SSL environment variable, can be used whether or not `mod_ssl` is
loaded, but will always expand to the empty string if it is not. Example: %{SSL:SSL_CIPHER_USEKEYSIZE} may expand to 128. These variables are available even without setting the StdEnvVars option of the SSLOptions directive.

4. %{HTTP:header}, where header can be any HTTP MIME-header name, can always be used to obtain the value of a header sent in the HTTP request. Example: %{HTTP:Proxy-Connection} is the value of the HTTP header ``Proxy-Connection:''.

If a HTTP header is used in a condition this header is added to the Vary header of the response in case the condition evaluates to true for the request. It is not added if the condition evaluates to false for the request. Adding the HTTP header to the Vary header of the response is needed for proper caching.

It has to be kept in mind that conditions follow a short circuit logic in the case of the 'ornext|OR' flag so that certain conditions might not be evaluated at all.

5. %{LA-U:variable} can be used for look-aheads which perform an internal (URL-based) sub-request to determine the final value of variable. This can be used to access variable for rewriting which is not available at the current stage, but will be set in a later phase.

For instance, to rewrite according to the REMOTE_USER variable from within the per-server context (httpd.conf file) you must use %{LA-U:REMOTE_USER} - this variable is set by the authorization phases, which come after the URL translation phase (during which mod_rewrite operates).

On the other hand, because mod_rewrite implements its per-directory context (.htaccess file) via the Fixup phase of the
API and because the authorization phases come before this phase, you just can use `\%{REMOTE_USER}` in that context.

6. `{LA-F:variable}` can be used to perform an internal (filename-based) sub-request, to determine the final value of `variable`. Most of the time, this is the same as LA-U above.

`CondPattern` is the condition pattern, a regular expression which is applied to the current instance of the `TestString`. `TestString` is first evaluated, before being matched against `CondPattern`.

**Remember:** `CondPattern` is a *perl compatible regular expression* with some additions:

1. You can prefix the pattern string with a `!' character (exclamation mark) to specify a **non**-matching pattern.

2. There are some special variants of `CondPatterns`. Instead of real regular expression strings you can also use one of the following:

   - `'<CondPattern'` (lexicographically precedes)  
     Treats the `CondPattern` as a plain string and compares it lexicographically to `TestString`. True if `TestString` lexicographically precedes `CondPattern`.

   - `'>CondPattern'` (lexicographically follows)  
     Treats the `CondPattern` as a plain string and compares it lexicographically to `TestString`. True if `TestString` lexicographically follows `CondPattern`.

   - `'=CondPattern'` (lexicographically equal)  
     Treats the `CondPattern` as a plain string and compares it lexicographically to `TestString`. True if `TestString` is lexicographically equal to `CondPattern` (the two strings are exactly equal, character for character). If `CondPattern` is `""` (two quotation marks) this compares `TestString` to


the empty string.

- `-d` (is directory)
  Treats the TestString as a pathname and tests whether or not it exists, and is a directory.

- `-f` (is regular file)
  Treats the TestString as a pathname and tests whether or not it exists, and is a regular file.

- `-s` (is regular file, with size)
  Treats the TestString as a pathname and tests whether or not it exists, and is a regular file with size greater than zero.

- `-l` (is symbolic link)
  Treats the TestString as a pathname and tests whether or not it exists, and is a symbolic link.

- `-x` (has executable permissions)
  Treats the TestString as a pathname and tests whether or not it exists, and has executable permissions. These permissions are determined according to the underlying OS.

- `-F` (is existing file, via subrequest)
  Checks whether or not TestString is a valid file, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!

- `-U` (is existing URL, via subrequest)
  Checks whether or not TestString is a valid URL, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!
This flag *only* returns information about things like access control, authentication, and authorization. This flag *does not* return information about the status code the configured handler (static file, CGI, proxy, etc.) would have returned.

**Note:**
All of these tests can also be prefixed by an exclamation mark ('!') to negate their meaning.

3. You can also set special flags for `CondPattern` by appending `[flags]` as the third argument to the `RewriteCond` directive, where `flags` is a comma-separated list of any of the following flags:

- `'nocase|NC' (no case)`
  This makes the test case-insensitive - differences between 'A-Z' and 'a-z' are ignored, both in the expanded `TestString` and the `CondPattern`. This flag is effective only for comparisons between `TestString` and `CondPattern`. It has no effect on filesystem and subrequest checks.

- `'ornext|OR' (or next condition)`
  Use this to combine rule conditions with a local OR instead of the implicit AND. Typical example:

  ```
  RewriteCond %{REMOTE_HOST} =host1 [OR]
  RewriteCond %{REMOTE_HOST} =host2 [OR]
  RewriteCond %{REMOTE_HOST} =host3
  RewriteRule ...some special stuff for any of these hosts...
  ```

  Without this flag you would have to write the condition/rule pair three times.

- `'novary|NV' (no vary)`
If a HTTP header is used in the condition, this flag prevents this header from being added to the Vary header of the response. Using this flag might break proper caching of the response if the representation of this response varies on the value of this header. So this flag should be only used if the meaning of the Vary header is well understood.

**Example:**

To rewrite the Homepage of a site according to the `"User-Agent:"` header of the request, you can use the following:

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla
RewriteRule ^/$ /homepage.max.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx
RewriteRule ^/$ /homepage.min.html [L]

RewriteRule ^/$ /homepage.std.html [L]
```

Explanation: If you use a browser which identifies itself as 'Mozilla' (including Netscape Navigator, Mozilla etc), then you get the max homepage (which could include frames, or other special features). If you use the Lynx browser (which is terminal-based), then you get the min homepage (which could be a version designed for easy, text-only browsing). If neither of these conditions apply (you use any other browser, or your browser identifies itself as something non-standard), you get the std (standard) homepage.
**Description:** Enables or disables runtime rewriting engine

**Syntax:**
RewriteEngine on|off

**Default:**
RewriteEngine off

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

**Override:**
FileInfo

**Status:**
Extension

**Module:**
mod_rewrite

The **RewriteEngine** directive enables or disables the runtime rewriting engine. If it is set to off this module does no runtime processing at all. It does not even update the SCRIPTInMillis environment variables.

Use this directive to disable the module instead of commenting out all the **RewriteRule** directives!

Note that rewrite configurations are not inherited by virtual hosts. This means that you need to have a **RewriteEngine** on directive for each virtual host in which you wish to use rewrite rules.

**RewriteMap** directives of the type prg are not started during server initialization if they're defined in a context that does not have **RewriteEngine** set to on
**Description:** Sets the name of the lock file used for RewriteMap synchronization

**Syntax:** RewriteLock `file-path`

**Context:** server config

**Status:** Extension

**Module:** mod_rewrite

This directive sets the filename for a synchronization lockfile which mod_rewrite needs to communicate with RewriteMap programs. Set this lockfile to a local path (not on a NFS-mounted device) when you want to use a rewriting map-program. It is not required for other types of rewriting maps.
**Description:** Sets the name of the file used for logging rewrite engine processing

**Syntax:** 

```
RewriteLog file-path|pipe
```

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_rewrite

The **RewriteLog** directive sets the name of the file to which the server logs any rewriting actions it performs. If the name does not begin with a slash ('/') then it is assumed to be relative to the **Server Root**. The directive should occur only once per server config.

To disable the logging of rewriting actions it is not recommended to set **Filename** to `/dev/null`, because although the rewriting engine does not then output to a logfile it still creates the logfile output internally. **This will slow down the server with no advantage to the administrator!** To disable logging either remove or comment out the **RewriteLog** directive or use **RewriteLogLevel 0**!

The **RewriteLog** log file format is as follows:

<table>
<thead>
<tr>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote host IP address</td>
<td>192.168.200.166</td>
</tr>
<tr>
<td>Remote login name</td>
<td>Will usually be &quot;-&quot;</td>
</tr>
<tr>
<td>HTTP user auth name</td>
<td>Username, or &quot;-&quot; if no auth</td>
</tr>
<tr>
<td>Date and time of request</td>
<td>[28/Aug/2009:13:09:09 --0400]</td>
</tr>
<tr>
<td>Virtualhost and virtualhost</td>
<td>[<a href="http://www.example.com/sid#84a650">www.example.com/sid#84a650</a>]</td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>Request ID, and whether</td>
<td>[rid#9f0e58/subreq]</td>
</tr>
<tr>
<td>it's a subrequest</td>
<td></td>
</tr>
<tr>
<td>Log entry severity level</td>
<td>(2)</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Text error message</td>
<td>forcing proxy-throughput with <a href="http://127.0.0.1:8080/index.html">http://127.0.0.1:8080/index.html</a></td>
</tr>
</tbody>
</table>

**Security**

See the [Apache Security Tips](#) document for details on how your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.

**Example**

```
# Log to a file:
RewriteLog "'/usr/local/var/apache/logs/rewrite.log"

# Log to a pipe:
RewriteLog "'|/path/to/parser.pl"
```
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Sets the verbosity of the log file used by the rewrite engine</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>RewriteLogLevel <em>Level</em></td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>RewriteLogLevel 0</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_rewrite</td>
</tr>
</tbody>
</table>

The **RewriteLogLevel** directive sets the verbosity level of the rewriting logfile. The default level 0 means no logging, while 9 or more means that practically all actions are logged.

To disable the logging of rewriting actions simply set *Level* to 0. This disables all rewrite action logs.

Using a high value for *Level* will slow down your Apache server dramatically! Use the rewriting logfile at a *Level* greater than 2 only for debugging!

**Example**

RewriteLogLevel 3
Definition: Defines a mapping function for key-lookup

Syntax: 
RewriteMap MapName MapType: MapSource

Context: server config, virtual host

Status: Extension

Module: mod_rewrite

Compatibility: The choice of different dbm types is available in Apache 2.0.41 and later

The **RewriteMap** directive defines a *Rewriting Map* which can be used inside rule substitution strings by the mapping-functions to insert/substitute fields through a key lookup. The source of this lookup can be of various types.

The *MapName* is the name of the map and will be used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

```
${ MapName : LookupKey }
${ MapName : LookupKey | DefaultValue }
```

When such a construct occurs, the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the key is not found then it is substituted by *DefaultValue* or by the empty string if no *DefaultValue* was specified. Empty values behave as if the key was absent, therefore it is not possible to distinguish between empty-valued keys and absent keys.

For example, you might define a **RewriteMap** as:

```
RewriteMap examplemap txt:/path/to/file/map.txt
```

You would then be able to use this map in a **RewriteRule** as
follows:

```
RewriteRule ^/ex/(.*) $examplemap:$1
```

The following combinations for MapType and MapSource can be used:

- **Standard Plain Text**
  MapType: `txt`, MapSource: Unix filesystem path to valid regular file
  This is the standard rewriting map feature where the MapSource is a plain ASCII file containing either blank lines, comment lines (starting with a '#' character) or pairs like the following - one per line.

  **MatchingKey SubstValue**

  **Example**

  ```
  ##
  ##    map.txt -- rewriting map
  ##
  Ralf.S.Engelschall  rse  # Bastard Operator From Hell
  Mr.Joe.Average      joe  # Mr. Average
  ```

  `RewriteMap real-to-user txt:/path/to/file/map.txt`

- **Randomized Plain Text**
  MapType: `rnd`, MapSource: Unix filesystem path to valid regular file
  This is identical to the Standard Plain Text variant above but with a special post-processing feature: After looking up a value it is parsed according to contained `\|` characters
which have the meaning of `or`. In other words they indicate a set of alternatives from which the actual returned value is chosen randomly. For example, you might use the following map file and directives to provide a random load balancing between several back-end servers, via a reverse-proxy. Images are sent to one of the servers in the 'static' pool, while everything else is sent to one of the 'dynamic' pool.

Example:

**Rewrite map file**

```plaintext
##
##  map.txt -- rewriting map
##

static  www1|www2|www3|www4
dynamic www5|www6
```

**Configuration directives**

```plaintext
RewriteMap servers rnd:/path/to/file/map.txt

RewriteRule ^/(.*\.(png|gif|jpg)) http://${servers:static}$/1 [NC,P,L]
RewriteRule ^/(.*) http://${servers:dynamic}$/1 [P,L]
```

- **Hash File**
  
  MapType: dbm[=type], MapSource: Unix filesystem path to valid regular file
  
  Here the source is a binary format DBM file containing the same contents as a *Plain Text* format file, but in a special representation which is optimized for really fast lookups. The *type* can be sdbm, gdbm, ndbm, or db depending on compile-time settings. If the *type* is omitted, the compile-time default will be chosen.
To create a dbm file from a source text file, use the `httxt2dbm` utility.

```
$ httxt2dbm -i mapfile.txt -o mapfile.map
```

- **Internal Function**
  MapType: `int`, MapSource: Internal Apache function
  Here, the source is an internal Apache function. Module authors can provide additional internal functions by registering them with the `ap_register_rewrite_mapfunc` API. The functions that are provided by default are:

  - `toupper`:
    Converts the key to all upper case.
  - `tolower`:
    Converts the key to all lower case.
  - `escape`:
    Translates special characters in the key to hex-encodings.
  - `unescape`:
    Translates hex-encodings in the key back to special characters.

- **External Rewriting Program**
  MapType: `prg`, MapSource: Unix filesystem path to valid regular file
  Here the source is a program, not a map file. To create it you can use a language of your choice, but the result has to be an executable program (either object-code or a script with the magic cookie trick `#!/path/to/interpreter` as the first line).

  This program is started once, when the Apache server is started, and then communicates with the rewriting engine via
its stdin and stdout file-handles. For each map-function lookup it will receive the key to lookup as a newline-terminated string on stdin. It then has to give back the looked-up value as a newline-terminated string on stdout or the four-character string `"NULL"` if it fails (i.e., there is no corresponding value for the given key). A trivial program which will implement a 1:1 map (i.e., key == value) could be:

External rewriting programs are not started if they're defined in a context that does not have RewriteEngine set to on.

```perl
#!/usr/bin/perl
$| = 1;
while (<STDIN>) {
    # ...put here any transformations or lookups...
    print $_;
}
```

But be very careful:

1. `"Keep it simple, stupid"` (KISS). If this program hangs, it will cause Apache to hang when trying to use the relevant rewrite rule.

2. A common mistake is to use buffered I/O on stdout. Avoid this, as it will cause a deadloop! `"$| =1"` is used above, to prevent this.

3. The RewriteLock directive can be used to define a lockfile which mod_rewrite can use to synchronize communication with the mapping program. By default no such synchronization takes place.

The RewriteMap directive can occur more than once. For each mapping-function use one RewriteMap directive to declare its
rewriting mapfile. While you cannot declare a map in per-directory context it is of course possible to use this map in per-directory context.

**Note**
For plain text and DBM format files the looked-up keys are cached in-core until the mtime of the mapfile changes or the server does a restart. This way you can have map-functions in rules which are used for every request. This is no problem, because the external lookup only happens once!
### Description:
Sets some special options for the rewrite engine

### Syntax:
`RewriteOptions Options`

### Context:
server config, virtual host, directory, .htaccess

### Override:
FileInfo

### Status:
Extension

### Module:
`mod_rewrite`

### Compatibility:
MaxRedirects is no longer available in version 2.1 and later

---

The `RewriteOptions` directive sets some special options for the current per-server or per-directory configuration. The `Option` string can currently only be one of the following:

**inherit**

This forces the current configuration to inherit the configuration of the parent. In per-virtual-server context, this means that the maps, conditions and rules of the main server are inherited. In per-directory context this means that conditions and rules of the parent directory's `.htaccess` configuration are inherited.

Rules inherited from the parent scope are applied *after* rules specified in the child scope.

**AllowAnyURI**

When `RewriteRule` is used in `VirtualHost` or server context with version 2.2.23 or later of `httpd`, `mod_rewrite` will only process the rewrite rules if the request URI is a URL-path. This avoids some security issues where particular rules could allow "surprising" pattern expansions (see [CVE-2011-3368](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3368) and [CVE-2011-4317](https://cve.mitre.org/cgi-bin/cvenAME.cgi?name=CVE-2011-4317)). To lift the restriction on matching a URL-path, the `AllowAnyURI` option can be enabled, and
mod_rewrite will apply the rule set to any request URI string, regardless of whether that string matches the URL-path grammar required by the HTTP specification.

**Security Warning**

Enabling this option will make the server vulnerable to security issues if used with rewrite rules which are not carefully authored. It is strongly recommended that this option is not used. In particular, beware of input strings containing the '@' character which could change the interpretation of the transformed URI, as per the above CVE names.

**MergeBase**

With this option, the value of RewriteBase is copied from where it's explicitly defined into any sub-directory or sub-location that doesn't define its own RewriteBase. Not copying was the default until 2.2.22. In version 2.2.23 copying was the default. The flag to explicitly control it is available for Apache HTTP Server 2.2.24 and later.
**Description:** Defines rules for the rewriting engine

**Syntax:**

```
RewriteRule Pattern Substitution [flags]
```

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

**Override:** FileInfo

**Status:** Extension

**Module:** mod_rewrite

The **RewriteRule** directive is the real rewriting workhorse. The directive can occur more than once, with each instance defining a single rewrite rule. The order in which these rules are defined is important - this is the order in which they will be applied at runtime.

**Pattern** is a perl compatible regular expression. On the first RewriteRule it is applied to the ( %-decoded) **URL-path** of the request; subsequent patterns are applied to the output of the last matched RewriteRule.

**What is matched?**

In **VirtualHost** context, The **Pattern** will initially be matched against the part of the URL after the hostname and port, and before the query string (e.g. "/app1/index.html").

In **Directory** and .htaccess context, the **Pattern** will initially be matched against the **filesystem** path, after removing the prefix that led the server to the current RewriteRule (e.g. "app1/index.html" or "index.html" depending on where the directives are defined).

If you wish to match against the hostname, port, or query string, use a **RewriteCond** with the %{HTTP_HOST}, % {SERVER_PORT}, or %{QUERY_STRING} variables
Per-directory Rewrites

- The rewrite engine may be used in .htaccess files and in <Directory> sections, with some additional complexity.
- To enable the rewrite engine in this context, you need to set "RewriteEngine On" and "Options FollowSymLinks" must be enabled. If your administrator has disabled override of FollowSymLinks for a user's directory, then you cannot use the rewrite engine. This restriction is required for security reasons.
- When using the rewrite engine in .htaccess files the per-directory prefix (which always is the same for a specific directory) is automatically removed for the RewriteRule pattern matching and automatically added after any relative (not starting with a slash or protocol name) substitution encounters the end of a rule set. See the RewriteBase directive for more information regarding what prefix will be added back to relative substitutions.
- If you wish to match against the full URL-path in a per-directory (htaccess) RewriteRule, use the % {REQUEST_URI} variable in a RewriteCond.
- The removed prefix always ends with a slash, meaning the matching occurs against a string which never has a leading slash. Therefore, a Pattern with ^/ never matches in per-directory context.
- Although rewrite rules are syntactically permitted in <Location> and <Files> sections, this should never be necessary and is unsupported.

For some hints on regular expressions, see the mod_rewrite Introduction.
In mod_rewrite, the NOT character (‘!’) is also available as a possible pattern prefix. This enables you to negate a pattern; to say, for instance: ``if the current URL does NOT match this pattern''. This can be used for exceptional cases, where it is easier to match the negative pattern, or as a last default rule.

**Note**

When using the NOT character to negate a pattern, you cannot include grouped wildcard parts in that pattern. This is because, when the pattern does NOT match (ie, the negation matches), there are no contents for the groups. Thus, if negated patterns are used, you cannot use $N in the substitution string!

The *Substitution* of a rewrite rule is the string that replaces the original URL-path that was matched by *Pattern*. The *Substitution* may be a:

**file-system path**

Designates the location on the file-system of the resource to be delivered to the client. Substitutions are only treated as a file-system path when the rule is configured in server (virtualhost) context and the first component of the path in the substitution is exists in the file-system.

**URL-path**

A [DocumentRoot]-relative path to the resource to be served. Note that [mod_rewrite] tries to guess whether you have specified a file-system path or a URL-path by checking to see if the first segment of the path exists at the root of the file-system. For example, if you specify a *Substitution* string of /www/file.html, then this will be treated as a URL-path unless a directory named www exists at the root or your file-system (or, in the case of using rewrites in a .htaccess file, relative to your document root), in which case it will be treated
as a file-system path. If you wish other URL-mapping directives (such as Alias) to be applied to the resulting URL-path, use the [PT] flag as described below.

**Absolute URL**

If an absolute URL is specified, mod_rewrite checks to see whether the hostname matches the current host. If it does, the scheme and hostname are stripped out and the resulting path is treated as a URL-path. Otherwise, an external redirect is performed for the given URL. To force an external redirect back to the current host, see the [R] flag below.

- (dash)

  A dash indicates that no substitution should be performed (the existing path is passed through untouched). This is used when a flag (see below) needs to be applied without changing the path.

In addition to plain text, the Substitution string can include:

1. back-references ($N) to the RewriteRule pattern
2. back-references (%N) to the last matched RewriteCond pattern
3. server-variables as in rule condition test-strings (% {VARNAME})
4. mapping-function calls (${mapname:key|default})

Back-references are identifiers of the form $N (N=0..9), which will be replaced by the contents of the Nth group of the matched Pattern. The server-variables are the same as for the TestString of a RewriteCond directive. The mapping-functions come from the RewriteMap directive and are explained there. These three types of variables are expanded in the order above.

Rewrite rules are applied to the results of previous rewrite rules, in
the order in which they are defined in the config file. The URL is **completely replaced** by the *Substitution* and the rewriting process continues until all rules have been applied, or it is explicitly terminated by a **L** flag, or other flag which implies immediate termination, such as **F**.

**Modifying the Query String**

By default, the query string is passed through unchanged. You can, however, create URLs in the substitution string containing a query string part. Simply use a question mark inside the substitution string to indicate that the following text should be re-injected into the query string. When you want to erase an existing query string, end the substitution string with just a question mark. To combine new and old query strings, use the **[QSA]** flag.

Additionally you can set special actions to be performed by appending **[flags]** as the third argument to the *RewriteRule* directive. *Flags* is a comma-separated list, surround by square brackets, of any of the flags in the following table. More details, and examples, for each flag, are available in the *Rewrite Flags* document.

<table>
<thead>
<tr>
<th>Flag and syntax</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Escape non-alphanumeric characters before transformation. [details ...]</td>
</tr>
<tr>
<td>chain</td>
<td>C</td>
</tr>
<tr>
<td>cookie</td>
<td>CO=NAME:VAL</td>
</tr>
<tr>
<td>discardpath</td>
<td>DPI</td>
</tr>
<tr>
<td>env</td>
<td>E=![!]</td>
</tr>
<tr>
<td>forbidden</td>
<td>F</td>
</tr>
<tr>
<td>gone</td>
<td>G</td>
</tr>
<tr>
<td>Handler</td>
<td>H=Content-handler</td>
</tr>
<tr>
<td>last</td>
<td>L</td>
</tr>
<tr>
<td>next</td>
<td>N</td>
</tr>
<tr>
<td>nocase</td>
<td>NC</td>
</tr>
<tr>
<td>noescape</td>
<td>NE</td>
</tr>
<tr>
<td>nosubreq</td>
<td>NS</td>
</tr>
<tr>
<td>proxy</td>
<td>P</td>
</tr>
<tr>
<td>passthrough</td>
<td>PT</td>
</tr>
<tr>
<td>qsappend</td>
<td>QSA</td>
</tr>
<tr>
<td>redirect</td>
<td>R[=code]</td>
</tr>
<tr>
<td>skip</td>
<td>S=num</td>
</tr>
</tbody>
</table>
**Home directory expansion**

When the substitution string begins with a string resembling "/~user" (via explicit text or backreferences), mod_rewrite performs home directory expansion independent of the presence or configuration of `mod_userdir`.

This expansion does not occur when the PT flag is used on the `RewriteRule` directive.

Here are all possible substitution combinations and their meanings:

**Inside per-server configuration (httpd.conf)**

for request ```GET /somepath/pathinfo```:  

<table>
<thead>
<tr>
<th>Given Rule</th>
<th>Resulting Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/somepath(.*) otherpath$1</code></td>
<td>invalid, not supported</td>
</tr>
<tr>
<td><code>/somepath(.*) otherpath$1 [R]</code></td>
<td>invalid, not supported</td>
</tr>
<tr>
<td><code>/somepath(.*) otherpath$1 [P]</code></td>
<td>invalid, not supported</td>
</tr>
<tr>
<td><code>/somepath(.*) /otherpath$1</code></td>
<td><code>/otherpath/pathinfo</code></td>
</tr>
<tr>
<td><code>/somepath(.*) /otherpath$1 [P]</code></td>
<td>doesn't make sense</td>
</tr>
<tr>
<td><code>/somepath(.*) </code><a href="http://thishost/otherpath$1%60">http://thishost/otherpath$1`</a></td>
<td><code>/otherpath/pathinfo</code></td>
</tr>
<tr>
<td><code>/somepath(.*) </code><a href="http://thishost/otherpath$1">http://thishost/otherpath$1</a> [P]`</td>
<td>doesn't make sense</td>
</tr>
</tbody>
</table>
Inside per-directory configuration for /somepath
(/physical/path/to/somepath/.htaccess, with
RewriteBase /somepath)
for request ``GET /somepath/localpath/pathinfo``:

<table>
<thead>
<tr>
<th>Given Rule</th>
<th>Resulting Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>^localpath(.*) otherpath$1</td>
<td>/somepath/otherpath$1</td>
</tr>
<tr>
<td>^localpath(.*) otherpath$1 [R]</td>
<td><a href="http://thishost/somepath/otherpath$1">http://thishost/somepath/otherpath$1</a></td>
</tr>
<tr>
<td>^localpath(.*) otherpath$1 [P]</td>
<td>doesn't make sense</td>
</tr>
<tr>
<td>^localpath(.*) /otherpath$1</td>
<td>/otherpath/pathinfo</td>
</tr>
<tr>
<td>^localpath(.*) /otherpath$1 [R]</td>
<td><a href="http://thishost/otherpath$1">http://thishost/otherpath$1</a></td>
</tr>
<tr>
<td>^localpath(.*) /otherpath$1 [P]</td>
<td>doesn't make sense</td>
</tr>
<tr>
<td>^localpath(.*) <a href="http://thishost/otherpath$1">http://thishost/otherpath$1</a></td>
<td>/otherpath/pathinfo</td>
</tr>
<tr>
<td>^localpath(.*) <a href="http://thishost/otherpath$1">http://thishost/otherpath$1</a> [P]</td>
<td>doesn't make sense</td>
</tr>
<tr>
<td>^localpath(.*) <a href="http://otherhost/otherpath$1">http://otherhost/otherpath$1</a></td>
<td>/otherpath/pathinfo</td>
</tr>
<tr>
<td>^localpath(.*) <a href="http://otherhost/otherpath$1">http://otherhost/otherpath$1</a> [P]</td>
<td>doesn't make sense</td>
</tr>
<tr>
<td>^localpath(.*) <a href="http://otherhost/otherpath$1">http://otherhost/otherpath$1</a></td>
<td><a href="http://otherhost/otherpath$1">http://otherhost/otherpath$1</a></td>
</tr>
<tr>
<td>(the [R] flag is redundant)</td>
<td></td>
</tr>
</tbody>
</table>
via internal pro
Apache mod_setenvif

mod_setenvif

BrowserMatch ^Mozilla netscape
BrowserMatch MSIE !netscape
BrowserMatch SetEnvIf User-Agent HT:

BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot

BrowserMatch ^Mozilla forms jpeg=yes browser=netscape
BrowserMatch "^Mozilla/[2-3]" tables agif frames javascript
BrowserMatch MSIE !javascript
BrowserMatchNoCase  regex \[!]env-variable\[=value\]  \[!]env-variable\[=value\]  

BrowserMatchNoCase  mac platform=macintosh  
BrowserMatchNoCase  win platform=windows  

BrowserMatch  BrowserMatchNoCase  
SetEnvIfNoCase  2:  

BrowserMatchNoCase  Robot is_a_robot  
SetEnvIfNoCase  User-Agent  Robot is_a_robot
SetEnvIf

1. HTTP ( [RFC 2616] ) Host, User-Agent, Referer, Accept-Language
2. :
   - Remote_Host - ()
   - Remote_Addr - IP
   - Server_Addr - IP (2.0.43)
   - Request_Method - ( GET, POST )
   - Request_Protocol -
   - Request_URI - URL
3. SetEnvIf SetEnvIf[NoCase] () ( regex ) regex attribute

1. varname
2. !varname
3. \texttt{varname=value}

"1"

\texttt{regex}

: 
\texttt{SetEnvIf Request\_URI "\.gif\$" object\_is\_image=gif}
\texttt{SetEnvIf Request\_URI "\.jpg\$" object\_is\_image=jpg}
\texttt{SetEnvIf Request\_URI "\.xbm\$" object\_is\_image=xbm}

:\ 
\texttt{SetEnvIf Referer www\.mydomain\.com intra\_site\_referral}

:\ 
\texttt{SetEnvIf object\_is\_image xbm XBIT\_PROCESSING=1}

:\ 
\texttt{SetEnvIf ^TS* ^[a-z].* HAVE\_TS}

\texttt{intra\_site\_referral}

"TS" [a-z]

- \texttt{Apache}
SetEnvIfNoCase attribute regex [!]env-variable[=value] [!]env-variable[=value]
...
SetEnvIfNoCase .htaccess
FileInfo
Base
mod_setenvif

SetEnvIfNoCase Host Apache.Org site=apache

HTTP Host: Apache.Org apache.org site "apache"
Apache mod_so

Unix ( .so )

Apache 1.3  Apache 2.0 — Apache
Apache 1.3.15  2.0  Windows
mod_so  ApacheModuleFoo.dll

Apache  API  UNIX  Windows
  Windows  Unix

Unix
Configure

Apache

DLL

AP_MODULE_DECLARE_DATA (Apache)

module foo_module;

module AP_MODULE_DECLARE_DATA foo_module;

Unix  Windows

DLL

libhttpd.lib .dsp

modules

DLL

modules
LoadFile

LoadFile filename [filename] ...

Extension

mod_so

LoadFile libexec/libxmlparse.so
LoadModule module filename

Extension mod_so

LoadModule status_module modules/mod_status.so

ServerRoot modules
Apache mod_speling

<table>
<thead>
<tr>
<th>URL</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>speling_module</td>
<td>mod_speling.c</td>
</tr>
</tbody>
</table>

Apache
CheckCaseOnly:

Limits the action of the spelling module to case corrections.

CheckCaseOnly on|off

CheckCaseOnly Off

.,.,.htaccess

Options

Extension

mod_speling
CheckSpelling

CheckSpelling on|off
CheckSpelling Off
,.,.htaccess
Options
Extension
mod_speling
CheckSpelling Apache 1.1 Apache 1.3
Apache Apache 1.3.2

DAV
mod_speling
doc34.html
Apache Module mod_ssl

| Description: | Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols |
| Status:      | Extension                                                   |
| Module Identifier: | ssl_module                                      |
| Source File: | mod_ssl.c                                                   |

Summary

This module provides SSL v2/v3 and TLS v1 support for the Apache HTTP Server. It was contributed by Ralf S. Engeschall based on his mod_ssl project and originally derived from work by Ben Laurie.

This module relies on [OpenSSL](https://www.openssl.org) to provide the cryptography engine.

Further details, discussion, and examples are provided in the [SSL documentation](https://httpd.apache.org/docs/).
This module can be configured to provide several items of SSL information as additional environment variables to the SSI and CGI namespace. This information is not provided by default for performance reasons. (See `SSLOptions StdEnvVars`, below.) The generated variables are listed in the table below. For backward compatibility the information can be made available under different names, too. Look in the `Compatibility` chapter for details on the compatibility variables.

<table>
<thead>
<tr>
<th>Variable Name:</th>
<th>Value Type:</th>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>flag</td>
<td>HTTPS is being used.</td>
</tr>
<tr>
<td>SSL_PROTOCOL</td>
<td>string</td>
<td>The SSL protocol version (SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2)</td>
</tr>
<tr>
<td>SSL_SESSION_ID</td>
<td>string</td>
<td>The hex-encoded SSL session id</td>
</tr>
<tr>
<td>SSL_CIPHER</td>
<td>string</td>
<td>The cipher specification name</td>
</tr>
<tr>
<td>SSL_CIPHER_EXPORT</td>
<td>string</td>
<td>true if cipher is an export cipher</td>
</tr>
<tr>
<td>SSL_CIPHER_USEKEYSIZE</td>
<td>number</td>
<td>Number of cipher bits (actually used)</td>
</tr>
<tr>
<td>SSL_CIPHER_ALGKEYSIZE</td>
<td>number</td>
<td>Number of cipher bits (possible)</td>
</tr>
<tr>
<td>SSL_COMPRESS_METHOD</td>
<td>string</td>
<td>SSL compression method negotiated</td>
</tr>
<tr>
<td>SSL_VERSION_INTERFACE</td>
<td>string</td>
<td>The mod_ssl program version</td>
</tr>
<tr>
<td>SSL_VERSION_LIBRARY</td>
<td>string</td>
<td>The OpenSSL program</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SSL_CLIENT_M_VERSION</td>
<td>string</td>
<td>The version of the client certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_M_SERIAL</td>
<td>string</td>
<td>The serial of the client certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_S_DN</td>
<td>string</td>
<td>Subject DN in client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_S_DN_x509</td>
<td>string</td>
<td>Component of client's Subject DN</td>
</tr>
<tr>
<td>SSL_CLIENT_I_DN</td>
<td>string</td>
<td>Issuer DN of client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_I_DN_x509</td>
<td>string</td>
<td>Component of client's Issuer DN</td>
</tr>
<tr>
<td>SSL_CLIENT_V_START</td>
<td>string</td>
<td>Validity of client's certificate (start time)</td>
</tr>
<tr>
<td>SSL_CLIENT_V_END</td>
<td>string</td>
<td>Validity of client's certificate (end time)</td>
</tr>
<tr>
<td>SSL_CLIENT_V_REMAIN</td>
<td>string</td>
<td>Number of days until client's certificate expires</td>
</tr>
<tr>
<td>SSL_CLIENT_A_SIG</td>
<td>string</td>
<td>Algorithm used for the signature of client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_A_KEY</td>
<td>string</td>
<td>Algorithm used for the public key of client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT</td>
<td>string</td>
<td>PEM-encoded client certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_CHAIN_n</td>
<td>string</td>
<td>PEM-encoded certificates in client certificate chain</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SSL_CLIENT_VERIFY</td>
<td>string</td>
<td>NONE, SUCCESS, GENEROUS or FAILED: reason</td>
</tr>
<tr>
<td>SSL_SERVER_M_VERSION</td>
<td>string</td>
<td>The version of the server certificate</td>
</tr>
<tr>
<td>SSL_SERVER_M_SERIAL</td>
<td>string</td>
<td>The serial of the server certificate</td>
</tr>
<tr>
<td>SSL_SERVER_S_DN</td>
<td>string</td>
<td>Subject DN in server's certificate</td>
</tr>
<tr>
<td>SSL_SERVER_S_DN_x509</td>
<td>string</td>
<td>Component of server's Subject DN</td>
</tr>
<tr>
<td>SSL_SERVER_I_DN</td>
<td>string</td>
<td>Issuer DN of server's certificate</td>
</tr>
<tr>
<td>SSL_SERVER_I_DN_x509</td>
<td>string</td>
<td>Component of server's Issuer DN</td>
</tr>
<tr>
<td>SSL_SERVER_V_START</td>
<td>string</td>
<td>Validity of server's certificate (start time)</td>
</tr>
<tr>
<td>SSL_SERVER_V_END</td>
<td>string</td>
<td>Validity of server's certificate (end time)</td>
</tr>
<tr>
<td>SSL_SERVER_A_SIG</td>
<td>string</td>
<td>Algorithm used for the signature of server's certificate</td>
</tr>
<tr>
<td>SSL_SERVER_A_KEY</td>
<td>string</td>
<td>Algorithm used for the public key of server's certificate</td>
</tr>
<tr>
<td>SSL_SERVER_CERT</td>
<td>string</td>
<td>PEM-encoded server certificate</td>
</tr>
<tr>
<td>SSL_TLS_SNI</td>
<td>string</td>
<td>Contents of the SNI TLS extension (if supplied with ClientHello)</td>
</tr>
</tbody>
</table>
x509 specifies a component of an X.509 DN; one of C, ST, L, O, OU, CN, T, I, G, S, D, UID, Email. In Apache 2.1 and later, x509 may also include a numeric _n suffix. If the DN in question contains multiple attributes of the same name, this suffix is used as an index to select a particular attribute. For example, where the server certificate subject DN included two OU fields, SSL_SERVER_S_DN_OU_0 and SSL_SERVER_S_DN_OU_1 could be used to reference each.

SSL_CLIENT_V_REMAIN is only available in version 2.1 and later.
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 you can find in the above table.

For backward compatibility there is additionally a special ``% {name}c'' cryptography format function provided. Information about this function is provided in the Compatibility chapter.

Example

``
CustomLog logs/ssl_request_log "%t %h %{SSL_PROTOCOL}x % {SSL_CIPHER}x "%r" %b"
``

These formats even work without setting the StdEnvVars option of the SSLOptions directive.
**Description:** File of concatenated PEM-encoded CA Certificates for Client Auth

**Syntax:**

SSLCACertificateFile file-path

**Context:**

server config, virtual host

**Status:**

Extension

**Module:**

mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to SSLCACertificatePath.

**Example**

SSLCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-client.crt
**Description:** Directory of PEM-encoded CA Certificates for Client Auth

**Syntax:** SSLCACertificatePath *directory-path*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose clients you deal with. These are used to verify the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named `hash-value.N`. And you should always make sure this directory contains the appropriate symbolic links.

**Example**

SSLCACertificatePath /usr/local/apache2/conf/ssl.crt/
Description: File of concatenated PEM-encoded CA Certificates for defining acceptable CA names

Syntax: SSLCADNRequestFile file-path

Context: server config, virtual host

Status: Extension

Module: mod_ssl

When a client certificate is requested by mod_ssl, a list of acceptable Certificate Authority names is sent to the client in the SSL handshake. These CA names can be used by the client to select an appropriate client certificate out of those it has available.

If neither of the directives SSLCADNRequestPath or SSLCADNRequestFile are given, then the set of acceptable CA names sent to the client is the names of all the CA certificates given by the SSLCACertificateFile and SSLCACertificatePath directives; in other words, the names of the CAs which will actually be used to verify the client certificate.

In some circumstances, it is useful to be able to send a set of acceptable CA names which differs from the actual CAs used to verify the client certificate - for example, if the client certificates are signed by intermediate CAs. In such cases, SSLCADNRequestPath and/or SSLCADNRequestFile can be used; the acceptable CA names are then taken from the complete set of certificates in the directory and/or file specified by this pair of directives.

SSLCADNRequestFile must specify an all-in-one file containing a concatenation of PEM-encoded CA certificates.
SSLCA DNRequestFile /usr/local/apache2/conf/ca-names.crt
**Description:** Directory of PEM-encoded CA Certificates for defining acceptable CA names

**Syntax:** SSLCADNRequestPath directory-path

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

This optional directive can be used to specify the set of acceptable CA names which will be sent to the client when a client certificate is requested. See the SSLCADNRequestFile directive for more details.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named hash-value.N. And you should always make sure this directory contains the appropriate symbolic links.

**Example**

SSLCADNRequestPath /usr/local/apache2/conf/ca-names.crt/
**Description:** File of concatenated PEM-encoded CA CRLs for Client Auth

**Syntax:** `SSLCARevocationFile file-path`

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose clients you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to `SSLCARevocationPath`.

**Example**

```
SSLCARevocationFile /usr/local/apache2/conf/ssl.crl/ca-bundle-client.crl
```
Description: Directory of PEM-encoded CA CRLs for Client Auth

Syntax: SSLCARevocationPath directory-path

Context: server config, virtual host

Status: Extension

Module: mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose clients you deal with. These are used to revoke the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named hash-value.rN. And you should always make sure this directory contains the appropriate symbolic links.

Example

SSLCARevocationPath /usr/local/apache2/conf/ssl.crl/
**Description:** File of PEM-encoded Server CA Certificates

**Syntax:** `SSLCertificateChainFile file-path`

**Context:** server config, virtual host

**Status:** Extension

**Module:** 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.

But be careful: Providing the certificate chain works only if you are using a *single* RSA or DSA based server certificate. If you are using a coupled RSA+DSA certificate pair, this will work only if actually both certificates use the *same* certificate chain. Else the browsers will be confused in this situation.

**Example**

```
SSLCertificateChainFile /usr/local/apache2/conf/ssl.crt/ca.crt
```
**Description:** Server PEM-encoded X.509 Certificate file

**Syntax:**
```plaintext
SSLCertificateFile file-path
```

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** ECC support is available in Apache 2.2.26 and later

This directive points to a file with certificate data in PEM format. At a minimum, the file must include an end-entity (leaf) certificate. The directive can be used up to three times (referencing different filenames) when an RSA, a DSA, and an ECC based server certificate is used in parallel.

Custom DH parameters and an EC curve name for ephemeral keys, can be added to end of the first file configured using `SSLCertificateFile`. This is supported in version 2.2.30 or later. Such parameters can be generated using the commands `openssl dhparam` and `openssl ecparam`. The parameters can be added as-is to the end of the first certificate file. Only the first file can be used for custom parameters, as they are applied independently of the authentication algorithm type.

Finally, the end-entity certificate's private key can also be added to the certificate file instead of using a separate `SSLCertificateKeyFile` directive. This practice is highly discouraged. If the private key is encrypted, the pass phrase dialog is forced at startup time.

**DH parameter interoperability with primes > 1024 bit**

Beginning with version 2.2.30, mod_ssl makes use of standardized DH parameters with prime lengths of 2048, 3072,
4096, 6144 and 8192 bits (from RFC 3526), and hands them out to clients based on the length of the certificate’s RSA/DSA key. With Java-based clients in particular (Java 7 or earlier), this may lead to handshake failures - see this FAQ answer for working around such issues.

Example

SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
**Description:** Server PEM-encoded Private Key file

**Syntax:** SSLCertificateKeyFile file-path

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** ECC support is available in Apache 2.2.26 and later

This directive points to the PEM-encoded private key file for the server. If the contained private key is encrypted, the pass phrase dialog is forced at startup time.

The directive can be used up to three times (referencing different filenames) when an RSA, a DSA, and an ECC based private key is used in parallel. For each SSLCertificateKeyFile directive, there must be a matching SSLCertificateFile directive.

The private key may also be combined with the certificate in the file given by SSLCertificateFile, but this practice is highly discouraged.

**Example**

SSLCertificateKeyFile
/usr/local/apache2/conf/ssl.key/server.key
**Description:** Cipher Suite available for negotiation in SSL handshake

**Syntax:** `SSLCipherSuite cipher-spec`

**Default:** `SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP`

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

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_ssl

This complex directive uses a colon-separated `cipher-spec` string consisting of OpenSSL cipher specifications to configure the Cipher Suite the client is permitted to negotiate in the SSL handshake phase. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured Cipher Suite after the HTTP request was read but before the HTTP response is sent.

An SSL cipher specification in `cipher-spec` is composed of 4 major attributes plus a few extra minor ones:

- **Key Exchange Algorithm:**
  RSA or Diffie-Hellman variants.

- **Authentication Algorithm:**
  RSA, Diffie-Hellman, DSS or none.

- **Cipher/Encryption Algorithm:**
  DES, Triple-DES, RC4, RC2, IDEA or none.

- **MAC Digest Algorithm:**
  MD5, SHA or SHA1.

An SSL cipher can also be an export cipher and is either a SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3). To
specify which ciphers to use, one can either specify all the Ciphers, one at a time, or use aliases to specify the preference and order for the ciphers (see Table 1).

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kRSA</td>
<td>RSA key exchange</td>
</tr>
<tr>
<td>kDHr</td>
<td>Diffie-Hellman key exchange with RSA key</td>
</tr>
<tr>
<td>kDHD</td>
<td>Diffie-Hellman key exchange with DSA key</td>
</tr>
<tr>
<td>kEDH</td>
<td>Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)</td>
</tr>
</tbody>
</table>

Key Exchange Algorithm:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aNULL</td>
<td>No authentication</td>
</tr>
<tr>
<td>aRSA</td>
<td>RSA authentication</td>
</tr>
<tr>
<td>aDSS</td>
<td>DSS authentication</td>
</tr>
<tr>
<td>aDH</td>
<td>Diffie-Hellman authentication</td>
</tr>
</tbody>
</table>

Authentication Algorithm:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eNULL</td>
<td>No encoding</td>
</tr>
<tr>
<td>DES</td>
<td>DES encoding</td>
</tr>
<tr>
<td>3DES</td>
<td>Triple-DES encoding</td>
</tr>
<tr>
<td>RC4</td>
<td>RC4 encoding</td>
</tr>
<tr>
<td>RC2</td>
<td>RC2 encoding</td>
</tr>
<tr>
<td>IDEA</td>
<td>IDEA encoding</td>
</tr>
</tbody>
</table>

Cipher Encoding Algorithm:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>MD5 hash function</td>
</tr>
<tr>
<td>SHA1</td>
<td>SHA1 hash function</td>
</tr>
<tr>
<td>SHA2</td>
<td>SHA hash function</td>
</tr>
</tbody>
</table>

MAC Digest Algorithm:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLv2</td>
<td>all SSL version 2.0 ciphers</td>
</tr>
<tr>
<td>SSLv3</td>
<td>all SSL version 3.0 ciphers</td>
</tr>
</tbody>
</table>

Aliases:
<table>
<thead>
<tr>
<th>Cipher</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSv1</td>
<td>all TLS version 1.0 ciphers</td>
</tr>
<tr>
<td>EXP</td>
<td>all export ciphers</td>
</tr>
<tr>
<td>EXPORT40</td>
<td>all 40-bit export ciphers only</td>
</tr>
<tr>
<td>EXPORT56</td>
<td>all 56-bit export ciphers only</td>
</tr>
<tr>
<td>LOW</td>
<td>all low strength ciphers (no export, single DES)</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>all ciphers with 128 bit encryption</td>
</tr>
<tr>
<td>HIGH</td>
<td>all ciphers using Triple-DES</td>
</tr>
<tr>
<td>RSA</td>
<td>all ciphers using RSA key exchange</td>
</tr>
<tr>
<td>DH</td>
<td>all ciphers using Diffie-Hellman key exchange</td>
</tr>
<tr>
<td>EDH</td>
<td>all ciphers using Ephemeral Diffie-Hellman key exchange</td>
</tr>
<tr>
<td>ADH</td>
<td>all ciphers using Anonymous Diffie-Hellman key exchange</td>
</tr>
<tr>
<td>DSS</td>
<td>all ciphers using DSS authentication</td>
</tr>
<tr>
<td>NULL</td>
<td>all ciphers using no encryption</td>
</tr>
</tbody>
</table>

Now where this becomes interesting is that these can be put together to specify the order and ciphers you wish to use. To speed this up there are also aliases (SSLv2, SSLv3, TLSv1, EXP, LOW, MEDIUM, HIGH) for certain groups of ciphers. These tags can be joined together with prefixes to form the cipher-spec. Available prefixes are:

- none: add cipher to list
- +: move matching ciphers to the current location in list
- -: remove cipher from list (can be added later again)
- !: kill cipher from list completely (can not be added later again)

**aNULL, eNULL and EXP ciphers are always disabled**

Beginning with version 2.2.30, null and export-grade ciphers are...
always disabled, as mod_ssl unconditionally prepends any supplied cipher suite string with !aNULL:!eNULL:!EXP: at initialization.

A simpler way to look at all of this is to use the `openssl ciphers -v` command which provides a nice way to successively create the correct cipher-spec string. The default cipher-spec string is
`
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.

The complete list of particular RSA & DH ciphers for SSL is given in Table 2.

### Example

**SSLCipherSuite**  RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW

<table>
<thead>
<tr>
<th>Cipher-Tag</th>
<th>Protocol</th>
<th>Key Ex.</th>
<th>Auth.</th>
<th>Enc.</th>
<th>MAC</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RSA Ciphers:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES-CBC3-SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>3DES(168)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Protocol</td>
<td>Key Length</td>
<td>Cipher</td>
<td>Hash</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>------------</td>
<td>--------</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES-CBC3-MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>3DES(168) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDEA-CBC-SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>IDEA(128) SHA1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC4-SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(128) SHA1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC4-MD5</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(128) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDEA-CBC-MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>IDEA(128) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC2-CBC-MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>RC2(128) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC4-MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(128) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES-CBC-SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>DES(56) SHA1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC4-64-MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(64) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES-CBC-MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>DES(56) MD5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>DES(40) SHA1 export</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP-RC2-CBC-MD5</td>
<td>SSLv3</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC2(40) MD5 export</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP-RC4-MD5</td>
<td>SSLv3</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC4(40) MD5 export</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP-RC2-CBC-MD5</td>
<td>SSLv2</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC2(40) MD5 export</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP-RC4-MD5</td>
<td>SSLv2</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC4(40) MD5 export</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NULL-SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>None SHA1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NULL-MD5</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>None MD5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diffie-Hellman Ciphers:**
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Version</th>
<th>Key</th>
<th>Initial</th>
<th>MAC</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADH-DES-CBC3-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>None</td>
<td>3DES(168)</td>
<td>SHA1</td>
</tr>
<tr>
<td>ADH-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>None</td>
<td>DES(56)</td>
<td>SHA1</td>
</tr>
<tr>
<td>ADH-RC4-MD5</td>
<td>SSLv3</td>
<td>DH</td>
<td>None</td>
<td>RC4(128)</td>
<td>MD5</td>
</tr>
<tr>
<td>EDH-RSA-DES-CBC3-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>RSA</td>
<td>3DES(168)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EDH-DSS-DES-CBC3-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>DSS</td>
<td>3DES(168)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EDH-RSA-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>RSA</td>
<td>DES(56)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EDH-DSS-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>DSS</td>
<td>DES(56)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EXP-EDH-RSA-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>RSA</td>
<td>DES(40)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EXP-EDH-DSS-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>DSS</td>
<td>DES(40)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EXP-ADH-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>None</td>
<td>DES(40)</td>
<td>SHA1</td>
</tr>
<tr>
<td>EXP-ADH-RC4-MD5</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>None</td>
<td>RC4(40)</td>
<td>MD5</td>
</tr>
</tbody>
</table>
**Description:** Enable compression on the SSL level

**Syntax:** SSLCompression on|off

**Default:** SSLCompression off

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Available in httpd 2.2.24 and later, if using OpenSSL 0.9.8 or later; virtual host scope available if using OpenSSL 1.0.0 or later. The default used to be on in versions 2.2.24 to 2.2.25.

This directive allows to enable compression on the SSL level.

Enabling compression causes security issues in most setups (the so called CRIME attack).
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Enable use of a cryptographic hardware accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>SSLCryptoDevice <em>engine</em></td>
</tr>
<tr>
<td><strong>Default:</strong></td>
<td>SSLCryptoDevice <em>builtin</em></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_ssl</td>
</tr>
<tr>
<td><strong>Compatibility:</strong></td>
<td>Available in Apache 2.1 and later, if using -engine flavor of OpenSSL 0.9.6, or OpenSSL 0.9.7 or later</td>
</tr>
</tbody>
</table>

This directive enables use of a cryptographic hardware accelerator board to offload some of the SSL processing overhead. This directive can only be used if the SSL toolkit is built with "engine" support; OpenSSL 0.9.7 and later releases have "engine" support by default, the separate "-engine" releases of OpenSSL 0.9.6 must be used.

To discover which engine names are supported, run the command "openssl engine".

**Example**

```bash
# For a Broadcom accelerator:
SSLCryptoDevice ubsec
```
This directive toggles the usage of the SSL/TLS Protocol Engine. This should be used inside a `<VirtualHost>` section to enable SSL/TLS for a that virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts.

**Example**

```html
<VirtualHost _default_:443>
  SSLEngine on
  ...
</VirtualHost>
```

In Apache 2.1 and later, `SSLEngine` can be set to `optional`. This enables support for [RFC 2817](https://tools.ietf.org/html/rfc2817), Upgrading to TLS Within HTTP/1.1. At this time no web browsers support RFC 2817.
**Description:** SSL FIPS mode Switch

**Syntax:** SSLFIPS on|off

**Default:** SSLFIPS off

**Context:** server config

**Status:** Extension

**Module:** mod_ssl

This directive toggles the usage of the SSL library FIPS_mode flag. It must be set in the global server context and cannot be configured with conflicting settings (SSLFIPS on followed by SSLFIPS off or similar). The mode applies to all SSL library operations.

If httpd was compiled against an SSL library which did not support the FIPS_mode flag, SSLFIPS on will fail. Refer to the FIPS 140-2 Security Policy document of the SSL provider library for specific requirements to use mod_ssl in a FIPS 140-2 approved mode of operation; note that mod_ssl itself is not validated, but may be described as using FIPS 140-2 validated cryptographic module, when all components are assembled and operated under the guidelines imposed by the applicable Security Policy.
**Description:** Option to prefer the server's cipher preference order

**Syntax:**
SSLHonorCipherOrder *flag*

**Context:**
server config, virtual host

**Status:**
Extension

**Module:**
mod_ssl

**Compatibility:**
Available in Apache 2.1 and later, if using OpenSSL 0.9.7 or later

When choosing a cipher during an SSLv3 or TLSv1 handshake, normally the client's preference is used. If this directive is enabled, the server's preference will be used instead.

**Example**
SSLHonorCipherOrder on
**Description:** Option to enable support for insecure renegotiation

**Syntax:** SSLInsecureRenegotiation *flag*

**Default:** SSLInsecureRenegotiation off

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Available in httpd 2.2.15 and later, if using OpenSSL 0.9.8m or later

As originally specified, all versions of the SSL and TLS protocols (up to and including TLS/1.2) were vulnerable to a Man-in-the-Middle attack ([CVE-2009-3555](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555)) during a renegotiation. This vulnerability allowed an attacker to "prefix" a chosen plaintext to the HTTP request as seen by the web server. A protocol extension was developed which fixed this vulnerability if supported by both client and server.

If [mod_ssl](https://httpd.apache.org/docs/current/mod/mod_ssl.html) is linked against OpenSSL version 0.9.8m or later, by default renegotiation is only supported with clients supporting the new protocol extension. If this directive is enabled, renegotiation will be allowed with old (unpatched) clients, albeit insecurely.

**Security warning**

If this directive is enabled, SSL connections will be vulnerable to the Man-in-the-Middle prefix attack as described in [CVE-2009-3555](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-3555).

**Example**

SSLInsecureRenegotiation on
The SSL_SECURE_RENEG environment variable can be used from an SSI or CGI script to determine whether secure renegotiation is supported for a given SSL connection.
**Description:** Semaphore for internal mutual exclusion of operations

**Syntax:** SSLMutex `type`

**Default:** SSLMutex `none`

**Context:** server config

**Status:** Extension

**Module:** mod_ssl

This configures the SSL engine's semaphore (aka. lock) which is used for mutual exclusion of operations which have to be done in a synchronized way between the pre-forked Apache server processes. This directive can only be used in the global server context because it's only useful to have one global mutex. This directive is designed to closely match the `AcceptMutex` directive.

The following Mutex *types* are available:

- **none | no**
  
  This is the default where no Mutex is used at all. Use it at your own risk. But because currently the Mutex is mainly used for synchronizing write access to the SSL Session Cache you can live without it as long as you accept a sometimes garbled Session Cache. So it's not recommended to leave this the default. Instead configure a real Mutex.

- **posixsem**
  
  This is an elegant Mutex variant where a Posix Semaphore is used when possible. It is only available when the underlying platform and APR supports it.

- **sysvsem**
  
  This is a somewhat elegant Mutex variant where a SystemV IPC Semaphore is used when possible. It is possible to "leak"
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.

- **fcnt1:/path/to/mutex**
  This is a portable Mutex variant where a physical (lock-)file and the fcnt1() 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 fcnt1:/path/to/mutex method with the exception that the flock() function is used to provide file locking. It is only available when the underlying platform and APR supports it.

- **file:/path/to/mutex**
This directive tells the SSL Module to pick the "best" file locking implementation available to it, choosing between `fcntl` and `flock`, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- **default | yes**
  This directive tells the SSL Module to pick the default locking implementation as determined by the platform and APR.

**Example**

SSLMutex file:/usr/local/apache/logs/ssl_mutex
This directive can be used to control various run-time options on a per-directory basis. Normally, if multiple SSLOptions could apply to a directory, then the most specific one is taken completely; the options are not merged. However if all the options on the SSLOptions directive are preceded by a plus (+) or minus (-) symbol, the options are merged. Any options preceded by a + are added to the options currently in force, and any options preceded by a - are removed from the options currently in force.

The available options are:

- **StdEnvVars**
  When this option is enabled, the standard set of SSL related CGI/SSI environment variables are created. This per default is disabled for performance reasons, because the information extraction step is a rather expensive operation. So one usually enables this option for CGI and SSI requests only.

- **ExportCertData**
  When this option is enabled, additional CGI/SSI environment variables are created: SSL_SERVER_CERT, SSL_CLIENT_CERT and SSL_CLIENT_CERT_CHAIN_n (with n = 0,1,2,...). These contain the PEM-encoded X.509 Certificates of server and client for the current HTTPS connection and can be used by CGI scripts for deeper Certificate checking. Additionally all other certificates of the
client certificate chain are provided, too. This bloats up the environment a little bit which is why you have to use this option to enable it on demand.

- **FakeBasicAuth**
  When this option is enabled, the Subject Distinguished Name (DN) of the Client X509 Certificate is translated into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is just the Subject of the Client's X509 Certificate (can be determined by running OpenSSL's openssl x509 command: openssl x509 -noout -subject -in certificate.crt). Note that no password is obtained from the user. Every entry in the user file needs this password: `"xxj31ZMTZzkVA"`, which is the DES-encrypted version of the word `"password"`. Those who live under MD5-based encryption (for instance under FreeBSD or BSD/OS, etc.) should use the following MD5 hash of the same word: `"$1$OXLyS...$Owx8s2/m9/gfkcRVXzgoE/"`.

- **StrictRequire**
  This *forces* forbidden access when SSLRequireSSL or SSLRequire successfully decided that access should be forbidden. Usually the default is that in the case where a ``Satisfy any`` directive is used, and other access restrictions are passed, denial of access due to SSLRequireSSL or SSLRequire is overridden (because that's how the Apache Satisfy mechanism should work.) But for strict access restriction you can use SSLRequireSSL and/or SSLRequire in combination with an ``SSLOptions +StrictRequire``. Then an additional ``Satisfy Any`` has no chance once mod_ssl has decided to deny access.

- **OptRenegotiate**
This enables optimized SSL connection renegotiation handling when SSL directives are used in per-directory context. By default a strict scheme is enabled where every per-directory reconfiguration of SSL parameters causes a full SSL renegotiation handshake. When this option is used mod_ssl tries to avoid unnecessary handshakes by doing more granular (but still safe) parameter checks. Nevertheless these granular checks sometimes maybe not what the user expects, so enable this on a per-directory basis only, please.

Example

SSLOptions +FakeBasicAuth -StrictRequire
<Files ~ "\.(cgi|shtml)$">
SSLOptions +StdEnvVars -ExportCertData
<Files>
### SSLPassPhraseDialog

**Description:** Type of pass phrase dialog for encrypted private keys

**Syntax:** SSLPassPhraseDialog type

**Default:** SSLPassPhraseDialog builtin

**Context:** server config

**Status:** Extension

**Module:** mod_ssl

When Apache starts up it has to read the various Certificate (see SSLCertificateFile) and Private Key (see SSLCertificateKeyFile) files of the SSL-enabled virtual servers. Because for security reasons the Private Key files are usually encrypted, mod_ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. This query can be done in two ways which can be configured by type:

- **builtin**
  
  This is the default where an interactive terminal dialog occurs at startup time just before Apache detaches from the terminal. Here the administrator has to manually enter the Pass Phrase for each encrypted Private Key file. Because a lot of SSL-enabled virtual hosts can be configured, the following reuse-scheme is used to minimize the dialog: When a Private Key file is encrypted, all known Pass Phrases (at the beginning there are none, of course) are tried. If one of those known Pass Phrases succeeds no dialog pops up for this particular Private Key file. If none succeeded, another Pass Phrase is queried on the terminal and remembered for the next round (where it perhaps can be reused).

  This scheme allows mod_ssl to be maximally flexible (because for N encrypted Private Key files you can use N different Pass Phrases - but then you have to enter all of
them, of course) while minimizing the terminal dialog (i.e. when you use a single Pass Phrase for all N Private Key files this Pass Phrase is queried only once).

- `|/path/to/program [args...]`
  This mode allows an external program to be used which acts as a pipe to a particular input device; the program is sent the standard prompt text used for the builtin mode on stdin, and is expected to write password strings on stdout. If several passwords are needed (or an incorrect password is entered), additional prompt text will be written subsequent to the first password being returned, and more passwords must then be written back.

- `exec:/path/to/program`
  Here an external program is configured which is called at startup for each encrypted Private Key file. It is called with two arguments (the first is of the form `servername:portnumber`, the second is either `RSA`, `DSA`, or `ECC`), which indicate for which server and algorithm it has to print the corresponding Pass Phrase to stdout. The intent is that this external program first runs security checks to make sure that the system is not compromised by an attacker, and only when these checks were passed successfully it provides the Pass Phrase.

  Both these security checks, and the way the Pass Phrase is determined, can be as complex as you like. Mod_ssl just defines the interface: an executable program which provides the Pass Phrase on stdout. Nothing more or less! So, if you're really paranoid about security, here is your interface. Anything else has to be left as an exercise to the administrator, because local security requirements are so different.
The reuse-algorithm above is used here, too. In other words: The external program is called only once per unique Pass Phrase.

Example
SSLPassPhraseDialog exec:/usr/local/apache/sbin/pp-filter
**Description:** Configure usable SSL protocol flavors

**Syntax:**
SSLProtocol 
[+] protocol ...

**Default:**
SSLProtocol all

**Context:**
server config, virtual host

**Override:**
Options

**Status:**
Extension

**Module:**
mod_ssl

This directive can be used to control the SSL protocol flavors mod_ssl should use when establishing its server environment. Clients then can only connect with one of the provided protocols.

The available (case-insensitive) *protocols* are:

- **SSLv2**
  This is the Secure Sockets Layer (SSL) protocol, version 2.0. It is the original SSL protocol as designed by Netscape Corporation. Though its use has been deprecated, because of weaknesses in the security of the protocol.

- **SSLv3**
  This is the Secure Sockets Layer (SSL) protocol, version 3.0, from the Netscape Corporation. It is the successor to SSLv2 and the predecessor to TLSv1. It's supported by almost all popular browsers.

- **TLSv1**
  This is the Transport Layer Security (TLS) protocol, version 1.0. It is the successor to SSLv3 and is defined in RFC 2246.

- **TLSv1.1** *(when using OpenSSL 1.0.1 and later)*
  A revision of the TLS 1.0 protocol, as defined in RFC 4346.
- **TLSv1.2 (when using OpenSSL 1.0.1 and later)**
  A revision of the TLS 1.1 protocol, as defined in [RFC 5246](https://tools.ietf.org/html/rfc5246).

- **All**
  This is a shortcut for ```+SSLv2  +SSLv3  +TLSv1``` or `- when using OpenSSL 1.0.1 and later - ```+SSLv2  +SSLv3  +TLSv1 +TLSv1.1  +TLSv1.2```, respectively.

### Example

```yaml
# enable SSLv3 and all available TLSv1 flavors, but not SSLv2
SSLProtocol All -SSLv2
```
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>File of concatenated PEM-encoded CA Certificates for Remote Server Auth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>SSLProxyCACertificateFile <em>file-path</em></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_ssl</td>
</tr>
</tbody>
</table>

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCACertificatePath](#).

**Example**

SSLProxyCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-remote-server.crt
**Description:** Directory of PEM-encoded CA Certificates for Remote Server Auth

**Syntax:**

```
SSLProxyCACertificatePath directory-path
```

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose remote servers you deal with. These are used to verify the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named `hash-value.N`. And you should always make sure this directory contains the appropriate symbolic links.

**Example**

```
SSLProxyCACertificatePath /usr/local/apache2/conf/ssl.crt/
```
<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>File of concatenated PEM-encoded CA CRLs for Remote Server Auth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>SSLProxyCARevocationFile <code>file-path</code></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config, virtual host</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td>mod_ssl</td>
</tr>
</tbody>
</table>

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCARevocationPath](#).

**Example**

```
SSLProxyCARevocationFile /usr/local/apache2/conf/ssl.crl/ca-bundle-remote-server.crl
```
**Description:** Directory of PEM-encoded CA CRLs for Remote Server Auth

**Syntax:**
SSLProxyCARevocationPath directory-path

**Context:**
server config, virtual host

**Status:**
Extension

**Module:**
mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose remote servers you deal with. These are used to revoke the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links.

**Example**
SSLProxyCARevocationPath /usr/local/apache2/conf/ssl.crl/
Description: Whether to check the remote server certificates CN field
Syntax: SSLProxyCheckPeerCN on|off
Default: SSLProxyCheckPeerCN off
Context: server config, virtual host
Status: Extension
Module: mod_ssl

This directive sets whether the remote server certificates CN field is compared against the hostname of the request URL. If both are not equal a 502 status code (Bad Gateway) is sent.

Example
SSLProxyCheckPeerCN on
**Description:** Whether to check if remote server certificate is expired

**Syntax:** SSLProxyCheckPeerExpire on|off

**Default:** SSLProxyCheckPeerExpire off

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

This directive sets whether it is checked if the remote server certificate is expired or not. If the check fails a 502 status code (Bad Gateway) is sent.

**Example**

SSLProxyCheckPeerExpire on
**Description:** Cipher Suite available for negotiation in SSL proxy handshake.

**Syntax:**
SSLProxyCipherSuite cipher-spec

**Default:**
SSLProxyCipherSuite

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

**Override:**
AuthConfig

**Status:**
Extension

**Module:**
mod_ssl

Equivalent to SSLCipherSuite, but for the proxy connection.
Please refer to [SSLCipherSuite](#) for additional information.
**Description:** SSL Proxy Engine Operation Switch

**Syntax:**
SSLProxyEngine on|off

**Default:**
SSLProxyEngine off

**Context:**
server config, virtual host

**Status:**
Extension

**Module:**
mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine for proxy. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for proxy usage in a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for proxy both for the main server and all configured virtual hosts.

Note that the SSLProxyEngine directive should not, in general, be included in a virtual host that will be acting as a forward proxy (using `<Proxy>` or `<ProxyRequest>` directives. SSLProxyEngine is not required to enable a forward proxy server to proxy SSL/TLS requests.

**Example**

```xml
<VirtualHost _default_:443>
  SSLProxyEngine on
  ...
</VirtualHost>
```
**Description:** File of concatenated PEM-encoded CA certificates to be used by the proxy for choosing a certificate

**Syntax:**

```
SSLProxyMachineCertificateChainFile filename
```

**Context:** server config

**Override:** Not applicable

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Available in Apache 2.2.23 and later

This directive sets the all-in-one file where you keep the certificate chain for all of the client certs in use. This directive will be needed if the remote server presents a list of CA certificates that are not direct signers of one of the configured client certificates.

This referenced file is simply the concatenation of the various PEM-encoded certificate files. Upon startup, each client certificate configured will be examined and a chain of trust will be constructed.

**Security warning**

If this directive is enabled, all of the certificates in the file will be trusted as if they were also in `SSLProxyCACertificateFile`.

**Example**

```
SSLProxyMachineCertificateChainFile
/usr/local/apache2/conf/ssl.crt/proxyCA.pem
```
### SSLProxyMachineCertificateFile

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>File of concatenated PEM-encoded client certificates and keys to be used by the proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td><code>SSLProxyMachineCertificateFile filename</code></td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>server config</td>
</tr>
<tr>
<td><strong>Override:</strong></td>
<td>Not applicable</td>
</tr>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module:</strong></td>
<td><code>mod_ssl</code></td>
</tr>
</tbody>
</table>

This directive sets the all-in-one file where you keep the certificates and keys used for authentication of the proxy server to remote servers.

This referenced file is simply the concatenation of the various PEM-encoded certificate files, in order of preference. Use this directive alternatively or additionally to `SSLProxyMachineCertificatePath`.

**Currently there is no support for encrypted private keys**

**Example**

```
SSLProxyMachineCertificateFile
/usr/local/apache2/conf/ssl.crt/proxy.pem
```
SSLProxyMachineCertificatePath

**Description:** Directory of PEM-encoded client certificates and keys to be used by the proxy

**Syntax:**

```
SSLProxyMachineCertificatePath
directory
```

**Context:** server config

**Override:** Not applicable

**Status:** Extension

**Module:** mod_ssl

This directive sets the directory where you keep the certificates and keys used for authentication of the proxy server to remote servers.

The files in this directory must be PEM-encoded and are accessed through hash filenames. Additionally, you must create symbolic links named `hash-value.N`. And you should always make sure this directory contains the appropriate symbolic links.

Currently there is no support for encrypted private keys

**Example**

```
SSLProxyMachineCertificatePath
/usr/local/apache2/conf/proxy.crt/
```
**Description:** Configure usable SSL protocol flavors for proxy usage

**Syntax:** SSLProxyProtocol [+-]protocol ...

**Default:** SSLProxyProtocol all

**Context:** server config, virtual host

**Override:** Options

**Status:** Extension

**Module:** mod_ssl

This directive can be used to control the SSL protocol flavors mod_ssl should use when establishing its server environment for proxy. It will only connect to servers using one of the provided protocols.

Please refer to [SSLProtocol](#) for additional information.
**Description:** Type of remote server Certificate verification

**Syntax:**
SSLProxyVerify level

**Default:**
SSLProxyVerify none

**Context:**
server config, virtual host

**Status:**
Extension

**Module:**
mod_ssl

When a proxy is configured to forward requests to a remote SSL server, this directive can be used to configure certificate verification of the remote server.

Note that even when certificate verification is enabled, mod_ssl does **not** check whether the commonName (hostname) attribute of the server certificate matches the hostname used to connect to the server. In other words, the proxy does not guarantee that the SSL connection to the backend server is "secure" beyond the fact that the certificate is signed by one of the CAs configured using the `SSLProxyCACertificatePath` and/or `SSLProxyCACertificateFile` directives. In order to get this check done please have a look at `SSLProxyCheckPeerCN` and `SSLProxyCheckPeerExpire` directives which are off by default.

The following levels are available for `level`:

- **none**: no remote server Certificate is required at all
- **optional**: the remote server may present a valid Certificate
- **require**: the remote server has to present a valid Certificate
- **optional_no_ca**: the remote server may present a valid Certificate but it need not to be (successfully) verifiable.
In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all servers and level **optional_no_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

**Example**

```
SSLProxyVerify require
```
**Description:** Maximum depth of CA Certificates in Remote Server Certificate verification

**Syntax:** 
SSLProxyVerifyDepth *number*

**Default:**
SSLProxyVerifyDepth 1

**Context:**
server config, virtual host

**Override:**
AuthConfig

**Status:**
Extension

**Module:**
mod_ssl

This directive sets how deeply mod_ssl should verify before deciding that the remote server does not have a valid certificate.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the remote server certificate. A depth of 0 means that self-signed remote server certificates are accepted only, the default depth of 1 means the remote server certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under SSLProxyCACertificatePath), etc.

**Example**
SSLProxyVerifyDepth 10
**Description:** Pseudo Random Number Generator (PRNG) seeding source

**Syntax:**
```
SSLRandomSeed context source [bytes]
```

**Context:** server config

**Status:** Extension

**Module:** mod_ssl

This configures one or more sources for seeding the Pseudo Random Number Generator (PRNG) in OpenSSL at startup time (context is startup) and/or just before a new SSL connection is established (context is connect). This directive can only be used in the global server context because the PRNG is a global facility.

The following `source` variants are available:

- **builtin**
  This is the always available builtin seeding source. Its usage consumes minimum CPU cycles under runtime and hence can be always used without drawbacks. The source used for seeding the PRNG contains of the current time, the current process id and (when applicable) a randomly chosen 1KB extract of the inter-process scoreboard structure of Apache. The drawback is that this is not really a strong source and at startup time (where the scoreboard is still not available) this source just produces a few bytes of entropy. So you should always, at least for the startup, use an additional seeding source.

- **file:/path/to/source**
  This variant uses an external file `/path/to/source` as the source for seeding the PRNG. When `bytes` is specified, only the first `bytes` number of bytes of the file form the entropy (and `bytes` is given to `/path/to/source` as the first
argument). When bytes is not specified the whole file forms the entropy (and 0 is given to /path/to/source as the first argument). Use this especially at startup time, for instance with an available /dev/random and/or /dev/urandom devices (which usually exist on modern Unix derivatives like FreeBSD and Linux).

But be careful: Usually /dev/random provides only as much entropy data as it actually has, i.e. when you request 512 bytes of entropy, but the device currently has only 100 bytes available two things can happen: On some platforms you receive only the 100 bytes while on other platforms the read blocks until enough bytes are available (which can take a long time). Here using an existing /dev/urandom is better, because it never blocks and actually gives the amount of requested data. The drawback is just that the quality of the received data may not be the best.

On some platforms like FreeBSD one can even control how the entropy is actually generated, i.e. by which system interrupts. More details one can find under rndcontrol(8) on those platforms. Alternatively, when your system lacks such a random device, you can use a tool like EGD (Entropy Gathering Daemon) and run its client program with the exec:/path/to/program/ variant (see below) or use egd:/path/to/egd-socket (see below).

- exec:/path/to/program
  This variant uses an external executable /path/to/program as the source for seeding the PRNG. When bytes is specified, only the first bytes number of bytes of its stdout contents form the entropy. When bytes is not specified, the entirety of the data produced on stdout form the entropy. Use this only at startup time when you need a
very strong seeding with the help of an external program (for instance as in the example above with the truerand utility you can find in the mod_ssl distribution which is based on the AT&T truerand library). Using this in the connection context slows down the server too dramatically, of course. So usually you should avoid using external programs in that context.

- egd:/path/to/egd-socket (Unix only)
  This variant uses the Unix domain socket of the external Entropy Gathering Daemon (EGD) (see http://www.lothar.com/tech/crypto/) to seed the PRNG. Use this if no random device exists on your platform.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLRandomSeed startup builtin</td>
</tr>
<tr>
<td>SSLRandomSeed startup file:/dev/random</td>
</tr>
<tr>
<td>SSLRandomSeed startup file:/dev/urandom 1024</td>
</tr>
<tr>
<td>SSLRandomSeed startup exec:/usr/local/bin/truerand 16</td>
</tr>
<tr>
<td>SSLRandomSeed connect builtin</td>
</tr>
<tr>
<td>SSLRandomSeed connect file:/dev/random</td>
</tr>
<tr>
<td>SSLRandomSeed connect file:/dev/urandom 1024</td>
</tr>
</tbody>
</table>
**Description:** Set the size for the SSL renegotiation buffer

**Syntax:** SSLRenegBufferSize \( \text{bytes} \)

**Default:** SSLRenegBufferSize 131072

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_ssl

If an SSL renegotiation is required in per-location context, for example, any use of [SSLVerifyClient](https://httpd.apache.org/docs/current/mod/mod_ssl.html#sslverifyclient) in a Directory or Location block, then `mod_ssl` must buffer any HTTP request body into memory until the new SSL handshake can be performed. This directive can be used to set the amount of memory that will be used for this buffer.

---

Note that in many configurations, the client sending the request body will be untrusted so a denial of service attack by consumption of memory must be considered when changing this configuration setting.

---

**Example**

SSLRenegBufferSize 262144
**Description:** Allow access only when an arbitrarily complex boolean expression is true

**Syntax:** SSLRequire *expression*

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod_ssl

This directive specifies a general access requirement which has to be fulfilled in order to allow access. It is a very powerful directive because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks.

The implementation of SSLRequire is not thread safe. Using SSLRequire inside .htaccess files on a threaded MPM may cause random crashes.

The *expression* must match the following syntax (given as a BNF grammar notation):

```plaintext
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
```
while for varname any variable from Table 3 can be used. Finally for funcname the following functions are available:

- **file(filename)**
  This function takes one string argument and expands to the contents of the file. This is especially useful for matching this contents against a regular expression, etc.

Notice that *expression* is first parsed into an internal machine representation and then evaluated in a second step. Actually, in Global and Per-Server Class context *expression* is parsed at startup time and at runtime only the machine representation is executed. For Per-Directory context, specifically in a .htaccess context, this is different: here *expression* has to be parsed and immediately executed for every request.

**Example**
SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)/- /
and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20 ) \nor %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/

The OID( ) function expects to find zero or more instances of the given OID in the client certificate, and compares the left-hand side string against the value of matching OID attributes. Every matching OID is checked, until a match is found.

*Standard CGI/1.0 and Apache variables:*

```
HTTP_USER_AGENT          PATH_INFO          AUTH.
HTTP_REFERER             QUERY_STRING       SERVICES
HTTP_COOKIE              REMOTE_HOST       API
HTTP_FORWARDED           REMOTE_IDENT      TIME
HTTP_HOST                IS_SUBREQ          TIME
HTTP_PROXY_CONNECTION    DOCUMENT_ROOT      TIME
HTTP_ACCEPT              SERVER_ADMIN       TIME
HTTP:headername          SERVER_NAME        TIME
THE_REQUEST              SERVER_PORT        TIME
REQUEST_METHOD           SERVER_PROTOCOL     TIME
REQUEST_SCHEME           REMOTE_ADDR        TIME
REQUEST_URI              REMOTE_USER        ENV:
REQUEST_FILENAME
```

*SSL-related variables:*

```
HTTPS                      SSL_CLIENT_M_VERSION   SSL.
SSL_PROTOCOL               SSL_CLIENT_M_SERIAL   SSL.
SSL_SESSION_ID             SSL_CLIENT_V_START   SSL.
SSL_CIPHER                 SSL_CLIENT_V_END     SSL.
SSL_CIPHER_EXPORT          SSL_CLIENT_S_DN     SSL.
SSL_CIPHER_ALGKEYSIZE      SSL_CLIENT_S_DN_C    SSL.
SSL_CIPHER_USEKEYSIZE      SSL_CLIENT_S_DN_ST   SSL.
SSL_VERSION_LIBRARY       SSL_CLIENT_S_DN_L     SSL.
SSL_CLIENT_S_DN_O
```
SSL_VERSION_INTERFACE  SSL_CLIENT_S_DN_OU  SSL
SSL_CLIENT_S_DN_CN  SSL
SSL_CLIENT_S_DN_T  SSL
SSL_CLIENT_S_DN_I  SSL
SSL_CLIENT_S_DN_G  SSL
SSL_CLIENT_S_DN_S  SSL
SSL_CLIENT_S_DN_D  SSL
SSL_CLIENT_S_DN_UID  SSL
SSL_CLIENT_S_DN_Email  SSL
SSL_CLIENT_I_DN  SSL
SSL_CLIENT_I_DN_C  SSL
SSL_CLIENT_I_DN_ST  SSL
SSL_CLIENT_I_DN_L  SSL
SSL_CLIENT_I_DN_O  SSL
SSL_CLIENT_I_DN_OU  SSL
SSL_CLIENT_I_DN_CN  SSL
SSL_CLIENT_I_DN_T  SSL
SSL_CLIENT_I_DN_I  SSL
SSL_CLIENT_I_DN_G  SSL
SSL_CLIENT_I_DN_S  SSL
SSL_CLIENT_I_DN_D  SSL
SSL_CLIENT_I_DN_UID  SSL
SSL_CLIENT_I_DN_Email  SSL
SSL_CLIENT_A_SIG  SSL
SSL_CLIENT_A_KEY  SSL
SSL_CLIENT_CERT  SSL
SSL_CLIENT_CERT_CHAIN_n
SSL_CLIENT_VERIFY  SSL
This directive forbids access unless HTTP over SSL (i.e. HTTPS) is enabled for the current connection. This is very handy inside the SSL-enabled virtual host or directories for defending against configuration errors that expose stuff that should be protected. When this directive is present all requests are denied which are not using SSL.

Example

SSLRequireSSL
**Description:** Type of the global/inter-process SSL Session Cache

**Syntax:** SSLSessionCache  *type*

**Default:** SSLSessionCache  *none*

**Context:** server config

**Status:** Extension

**Module:** mod_ssl

This configures the storage type of the global/inter-process SSL Session Cache. This cache is an optional facility which speeds up parallel request processing. For requests to the same server process (via HTTP keep-alive), OpenSSL already caches the SSL session information locally. But because modern clients request inlined images and other data via parallel requests (usually up to four parallel requests are common) those requests are served by *different* pre-forked server processes. Here an inter-process cache helps to avoid unnecessary session handshakes.

The following four storage *types* are currently supported:

- **none**
  This disables the global/inter-process Session Cache. This will incur a noticeable speed penalty and may cause problems if using certain browsers, particularly if client certificates are enabled. This setting is not recommended.

- **nonenotnull**
  This disables any global/inter-process Session Cache. However it does force OpenSSL to send a non-null session ID to accommodate buggy clients that require one.

- **dbm:/path/to/datafile**
  This makes use of a DBM hashfile on the local disk to
synchronize the local OpenSSL memory caches of the server processes. This session cache may suffer reliability issues under high load.

- **shm:/path/to/datafile[(size)]**
  This makes use of a high-performance cyclic buffer (approx. size bytes in size) inside a shared memory segment in RAM (established via /path/to/datafile) to synchronize the local OpenSSL memory caches of the server processes. This is the recommended session cache.

- **dc:UNIX:/path/to/socket**
  This makes use of the distcache distributed session caching libraries. The argument should specify the location of the server or proxy to be used using the distcache address syntax; for example, UNIX:/path/to/socket specifies a UNIX domain socket (typically a local dc_client proxy); IP:server.example.com:9001 specifies an IP address.

### Examples

SSLSessionCache dbm:/usr/local/apache/logs/ssl_gcache_data
SSLSessionCache
shm:/usr/local/apache/logs/ssl_gcache_data(512000)
**Description:** Number of seconds before an SSL session expires in the Session Cache

**Syntax:**
`SSLSessionCacheTimeout seconds`

**Default:**
`SSLSessionCacheTimeout 300`

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Applies also to RFC 5077 TLS session resumption in Apache 2.2.28 and later

This directive sets the timeout in seconds for the information stored in the global/inter-process SSL Session Cache, the OpenSSL internal memory cache and for sessions resumed by TLS session resumption (RFC 5077). It can be set as low as 15 for testing, but should be set to higher values like 300 in real life.

**Example**
`SSLSessionCacheTimeout 600`
**Description:** Persistent encryption/decryption key for TLS session tickets

**Syntax:**

```
SSLSessionTicketKeyFile file-path
```

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Available in httpd 2.2.30 and later, if using OpenSSL 0.9.8h or later

Optionally configures a secret key for encrypting and decrypting TLS session tickets, as defined in [RFC 5077](https://tools.ietf.org/html/rfc5077). Primarily suitable for clustered environments where TLS sessions information should be shared between multiple nodes. For single-instance httpd setups, it is recommended to *not* configure a ticket key file, but to rely on (random) keys generated by mod_ssl at startup, instead.

The ticket key file must contain 48 bytes of random data, preferrably created from a high-entropy source. On a Unix-based system, a ticket key file can be created as follows:

```
dd if=/dev/random of=/path/to/file.tkey bs=1 count=48
```

Ticket keys should be rotated (replaced) on a frequent basis, as this is the only way to invalidate an existing session ticket - OpenSSL currently doesn't allow to specify a limit for ticket lifetimes. A new ticket key only gets used after restarting the web server. All existing session tickets become invalid after a restart.

The ticket key file contains sensitive keying material and should be protected with file permissions similar to those used for SSLCertificateKeyFile.
**Description:** Enable or disable use of TLS session tickets

**Syntax:** SSLSessionTickets on|off

**Default:** SSLSessionTickets on

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Available in httpd 2.2.30 and later, if using OpenSSL 0.9.8f or later.

This directive allows to enable or disable the use of TLS session tickets (RFC 5077).

TLS session tickets are enabled by default. Using them without restarting the web server with an appropriate frequency (e.g. daily) compromises perfect forward secrecy.
**Description:** Whether to allow non SNI clients to access a name based virtual host.

**Syntax:** SSLStrictSNIVHostCheck on|off

**Default:** SSLStrictSNIVHostCheck off

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_ssl

**Compatibility:** Available in Apache 2.2.12 and later

This directive sets whether a non SNI client is allowed to access a name based virtual host. If set to on in the non default name based virtual host, non SNI clients are not allowed to access this particular virtual host. If set to on in the default name based virtual host, non SNI clients are not allowed to access any name based virtual host belonging to this IP / port combination.

This option is only available if httpd was compiled against an SNI capable version of OpenSSL.

**Example**

SSLStrictSNIVHostCheck on
Description: Variable name to determine user name

Syntax: SSLUserName varname

Context: server config, directory, .htaccess

Override: AuthConfig

Status: Extension

Module: mod_ssl

Compatibility: Available in Apache 2.0.51 and later

This directive sets the "user" field in the Apache request object. This is used by lower modules to identify the user with a character string. In particular, this may cause the environment variable REMOTE_USER to be set. The varname can be any of the SSL environment variables.

Note that this directive has no effect if the FakeBasicAuth option is used (see SSLOptions).

Example

SSLUserName SSL_CLIENT_S_DN_CN
This directive sets the Certificate verification level for the Client Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification level after the HTTP request was read but before the HTTP response is sent.

The following levels are available for `level`:

- **none**: no client Certificate is required at all
- **optional**: the client *may* present a valid Certificate
- **require**: the client *has to* present a valid Certificate
- **optional_no_ca**: the client may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all browsers and level **optional_no_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

**Example**

```
SSLVerifyClient require
```
**Description:** Maximum depth of CA Certificates in Client Certificate verification

**Syntax:**

SSLVerifyDepth number

**Default:**

SSLVerifyDepth 1

**Context:**

server config, virtual host, directory, .htaccess

**Override:**

AuthConfig

**Status:**

Extension

**Module:**

mod_ssl

This directive sets how deeply mod_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the client certificate. A depth of 0 means that self-signed client certificates are accepted only, the default depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under SSLCACertificatePath), etc.

**Example**

SSLVerifyDepth 10
Apache mod_status

Status

"(*)"  ExtendedStatus  On
example.com

<Location /server-status>
SetHandler server-status

Order Deny, Allow
Deny from all
Allow from .example.com
</Location>

http://your.server.name
status?refresh=N
http://your.server.name/server-status?auto
Apache /support

mod_status
<table>
<thead>
<tr>
<th>Base</th>
<th>mod_status</th>
<th>ExtendedStatus Apache 1.3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ExtendedStatus Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ExtendedStatus On</td>
</tr>
</tbody>
</table>
ExtendedStatus On mod_status

63( 63 )63

Apache
GET /disk1/storage/apache/htdocs/images/imagestore1/food/apples.jpg
mod_status:

Off

GET /disk1/storage/apache/htdocs/images/imagestore1/food/fruits

On

GET /disk1/storage/apache/htdocs/images/imagestore1/food/apples.jpg

ExtendedStatus On mod_status

63( 63 )63

Apache
GET /disk1/storage/apache/htdocs/images/imagestore1/food/apples.jpg
mod_status:
# Apache Module mod_substitute

<table>
<thead>
<tr>
<th><strong>Description:</strong></th>
<th>Perform search and replace operations on response bodies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong></td>
<td>Extension</td>
</tr>
<tr>
<td><strong>Module Identifier:</strong></td>
<td>substitute_module</td>
</tr>
<tr>
<td><strong>Source File:</strong></td>
<td>mod_substitute.c</td>
</tr>
<tr>
<td><strong>Compatibility:</strong></td>
<td>Available in Apache 2.2.7 and later</td>
</tr>
</tbody>
</table>

## Summary

`mod_substitute` provides a mechanism to perform both regular expression and fixed string substitutions on response bodies.
The **Substitute** directive specifies a search and replace pattern to apply to the response body.

The meaning of the pattern can be modified by using any combination of these flags:

- **i**
  - Perform a case-insensitive match.
- **n**
  - By default the pattern is treated as a regular expression. Using the n flag forces the pattern to be treated as a fixed string.
- **f**
  - The f flag causes mod_substitute to flatten the result of a substitution allowing for later substitutions to take place on the boundary of this one. This is the default.
- **q**
  - The q flag causes mod_substitute to not flatten the buckets after each substitution. This can result in much faster response and a decrease in memory utilization, but should only be used if there is no possibility that the result of one substitution will ever match a pattern or regex of a subsequent one.
If either the pattern or the substitution contain a slash character then an alternative delimiter should be used:

**Example of using an alternate delimiter**

```xml
<Location />
    AddOutputFilterByType SUBSTITUTE text/html
    Substitute "s|<BR */?>|<br >/|i"
</Location>
```
**Description:** Change the merge order of inherited patterns

**Syntax:** SubstituteInheritBefore on|off

**Default:** SubstituteInheritBefore off

**Context:** directory, .htaccess

**Override:** FileInfo

**Status:** Extension

**Module:** mod_substitute

**Compatibility:** Available in httpd 2.2.32 and later

Whether to apply the inherited Substitute patterns first (on), or after the ones of the current context (off).

SubstituteInheritBefore is itself inherited, hence contexts that inherit it (those that don’t specify their own SubstituteInheritBefore value) will apply the closest defined merge order.
Apache mod_suexec

: CGI
: Extension
: suexec_module
: mod_suexec.c
: Apache 2.0

suexec CGI

SuEXEC
SuexecUserGroup CGI

1.3 VirtualHosts User Group

SuexecUserGroup nobody nogroup
Apache mod_unique_id

Extension
unique_id_module
mod_unique_id.c
Apache Unix

(NTP )

- NTP
- IP

pid ( ID) 32

httpd

Unix (UTC 1970 1 1 ) 16

( ip_addr, pid, time_stamp, counter ) httpd 65536

pid

httpd ( ÷ 10) modulo 65536 0)

pid pid

rand () seed seed

? 500

1.5%

UTC

NTP UTC

UNIQUE_ID 112 (32 IP 32 pid, 32 16

[A-Za-z0-9@-] MIME base64 19

base64 [A-Za-z0-9+/] + / URL
UNIQUE_ID

UNIQUE_ID

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0
Apache mod_userdir

http://example.com/~user/

URL
public_html
UserDir

- disabled enabled ()
- disabled
- enabled

enabled disabled UserDir
http://www.foo.com/~bob/one/two.html:

UserDir
UserDir public_html ~bob/public_html/one/two.html
UserDir /usr/web /usr/web/bob/one/two.html
UserDir /home/*/www /home/bob/www/one/two.html

UserDir
UserDir http://www.foo.com/users/bob/one/two.html
http://www.foo.com/users
UserDir http://www.foo.com/bob/usr/one/two.html
http://www.foo.com/*/usr
UserDir http://www.foo.com/~bob/one/two.html
http://www.foo.com/~*/

; "UserDir ./" "/~ro
**disabled root**" 

UserDir:

UserDir disabled
UserDir enabled user1 user2 user3

UserDir:

UserDir enabled
UserDir disabled user4 user5 user6


Userdir public_html /usr/web http://www.foo.com/

http://www.foo.com/~bob/one/two.html
~bob/public_html/one/two.html /usr/web/bob/one/two.html
http://www.foo.com/bob/one/two.html

Apache

2.1.4 UserDir UserDir public_html

- public_html
Apache Module mod_usertrack

**Description:** Clickstream logging of user activity on a site

**Status:** Extension

**Module Identifier:** usertrack_module

**Source File:** mod_usertrack.c

**Summary**

Previous releases of Apache have included a module which generates a 'clickstream' log of user activity on a site using cookies. This was called the "cookies" module, mod_cookies. In Apache 1.2 and later this module has been renamed the "user tracking" module, mod_usertrack. This module has been simplified and new directives added.
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}` in the log file format. For example:

```
CustomLog logs/clickstream "%{cookie} %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 might be useful.

True, Netscape claims that the correct format NOW is four digit dates, but four digit dates do in fact work... for Netscape Communicator 4.x, that is. However, 3.x and below do NOT accept them. It seems that Netscape originally had a 2-digit standard, and then with all of the Y2K hype and probably a few complaints, changed to a four digit format. Fortunately, 4.x also understands the 2-digit format. To ensure that your expiration date is legible to the client's browser, use 2-digit dates.

However, this does not limit expiration dates to the year 2000; if you use an expiration year of "13", for example, it is interpreted as 2013, not 1913! In fact, you can use an expiration year of up to "37", understood as "2037" by both MSIE and Netscape versions 3.x and up (not sure about versions previous to those). Not sure why Netscape chose a particular year as its cut-off point, but my guess is in respect to UNIX's 2038 problem. Netscape/MSIE 4.x seem to understand 2-digit years beyond that, at least until "50" for sure (they understand up until about "70", but not for sure).

Summary: Mozilla 3.x and up understands two digit years up until 2037. Mozilla 4.x understands up until at least 2050 in 2-digit format, but also understands 4-digit years, which can be up to 9999. Your best bet for sending a long-life cookie is to send it for some time late in the year "37".
**Description:** The domain to which the tracking cookie applies

**Syntax:** CookieDomain `domain`

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

**Override:** FileInfo

**Status:** Extension

**Module:** mod_usertrack

This directive controls the setting of the domain to which the tracking cookie applies. If not present, no domain is included in the cookie header field.

The domain string **must** begin with a dot, and **must** include at least one embedded dot. That is, `.example.com` is legal, but `foo.example.com` and `.com` are not.

Most browsers in use today will not allow cookies to be set for a two-part top level domain, such as `.co.uk`, although such a domain ostensibly fulfills the requirements above. These domains are equivalent to top level domains such as `.com`, and allowing such cookies may be a security risk. Thus, if you are under a two-part top level domain, you should still use your actual domain, as you would with any other top level domain (for example, use `.foo.co.uk`).
Description: Expiry time for the tracking cookie
Syntax: CookieExpires expiry-period
Context: server config, virtual host, directory, .htaccess
Override: FileInfo
Status: Extension
Module: mod_usertrack

When used, this directive sets an expiry time on the cookie generated by the usertrack module. The *expiry-period* can be given either as a number of seconds, or in the format such as "2 weeks 3 days 7 hours". Valid denominations are: years, months, weeks, days, hours, minutes and seconds. If the expiry time is in any format other than one number indicating the number of seconds, it must be enclosed by double quotes.

If this directive is not used, cookies last only for the current browser session.
**Description:** Name of the tracking cookie

**Syntax:** CookieName *token*

**Default:** CookieName Apache

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

**Override:** FileInfo

**Status:** Extension

**Module:** mod_usertrack

This directive allows you to change the name of the cookie this module uses for its tracking purposes. By default the cookie is named "Apache".

You must specify a valid cookie name; results are unpredictable if you use a name containing unusual characters. Valid characters include A-Z, a-z, 0-9, ",_", and ",-".
### Description
Format of the cookie header field

### Syntax
`CookieStyle`  
*Netscape|Cookie|Cookie2|RFC2109|RFC2965*

### Default
`CookieStyle Netscape`

### Context
`server config, virtual host, directory, .htaccess`

### Override
`FileInfo`

### Status
`Extension`

### Module
`mod_usertrack`

This directive controls the format of the cookie header field. The three formats allowed are:

- **Netscape**, which is the original but now deprecated syntax. This is the default, and the syntax Apache has historically used.

- **Cookie** or **RFC2109**, which is the syntax that superseded the Netscape syntax.

- **Cookie2** or **RFC2965**, which is the most current cookie syntax.

Not all clients can understand all of these formats, but you should use the newest one that is generally acceptable to your users' browsers. At the time of writing, most browsers only fully support `CookieStyle Netscape`.  

![Image](image-url)
**CookieTracking Directive**

**Description:** Enables tracking cookie

**Syntax:** CookieTracking on|off

**Default:** CookieTracking off

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

**Override:** FileInfo

**Status:** Extension

**Module:** mod_usertrack

When [mod_usertrack](https://httpd.apache.org/docs/current/mod/mod_usertrack.html) is loaded, and CookieTracking on is set, Apache will send a user-tracking cookie for all new requests. This directive can be used to turn this behavior on or off on a per-server or per-directory basis. By default, enabling [mod_usertrack](https://httpd.apache.org/docs/current/mod/mod_usertrack.html) will not activate cookies.
Apache HTTP 2.2
Apache mod_version

Extension
version_module
mod_version.c
Apache 2.0.56

httpd

<IfVersion 2.1.0>
   # current httpd version is exactly 2.1.0
</IfVersion>

<IfVersion >= 2.2>
   # use really new features :-)
</IfVersion>
<IfVersion> 
<IfVersion [[[!]operator] version]> ... 
</IfVersion> 
, , .htaccess 
All 
Extension 
mod_version

<IfVersion> 
httpd

major[.minor[.patch]] 2.1.0 2.2
patch 0

<table>
<thead>
<tr>
<th>operator</th>
<th>httpd</th>
</tr>
</thead>
<tbody>
<tr>
<td>= ==</td>
<td>httpd</td>
</tr>
<tr>
<td>&gt;</td>
<td>httpd</td>
</tr>
<tr>
<td>&gt;=</td>
<td>httpd</td>
</tr>
<tr>
<td>&lt;</td>
<td>httpd</td>
</tr>
<tr>
<td>&lt;=</td>
<td>httpd</td>
</tr>
</tbody>
</table>

<IfVersion >= 2.1>
   # this happens only in versions greater or
   # equal 2.1.0.
</IfVersion>

http
<table>
<thead>
<tr>
<th><strong>operator</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>= or ==</td>
<td>version</td>
</tr>
<tr>
<td></td>
<td>/regex/</td>
</tr>
<tr>
<td>~</td>
<td>version regex</td>
</tr>
</tbody>
</table>

```xml
<IfVersion = /^2.1.[01234]$/>  
    # e.g. workaround for buggy versions  
</IfVersion>

(   !):  

```xml
<IfVersion != ~ ^2.1.[01234]$>  
    # not for those versions  
</IfVersion>

**operator** =
Apache Module mod_vhost_alias

| Description: | Provides for dynamically configured mass virtual hosting |
| Status: | Extension |
| Module Identifier: | vhost_alias_module |
| Source File: | mod_vhost_alias.c |

Summary

This module creates dynamically configured virtual hosts, by allowing the IP address and/or the Host: header of the HTTP request to be used as part of the pathname to determine what files to serve. This allows for easy use of a huge number of virtual hosts with similar configurations.

Note

If mod_alias or mod_userdir are used for translating URIs to filenames, they will override the directives of mod_vhost_alias described below. For example, the following configuration will map /cgi-bin/script.pl to /usr/local/apache2/cgi-bin/script.pl in all cases:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
VirtualScriptAlias /never/found/%0/cgi-bin/
```

See also

- UseCanonicalName
- Dynamically configured mass virtual hosting
All the directives in this module interpolate a string into a pathname. The interpolated string (henceforth called the "name") may be either the server name (see the UseCanonicalName directive for details on how this is determined) or the IP address of the virtual host on the server in dotted-quad format. The interpolation is controlled by specifiers inspired by printf which have a number of formats:

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%%</td>
<td>insert a %</td>
</tr>
<tr>
<td>%p</td>
<td>insert the port number of the virtual host</td>
</tr>
<tr>
<td>%N.M</td>
<td>insert (part of) the name</td>
</tr>
</tbody>
</table>

N and M are used to specify substrings of the name. N selects from the dot-separated components of the name, and M selects characters within whatever N has selected. M is optional and defaults to zero if it isn't present; the dot must be present if and only if M is present. The interpretation is as follows:

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>the whole name</td>
</tr>
<tr>
<td>1</td>
<td>the first part</td>
</tr>
<tr>
<td>2</td>
<td>the second part</td>
</tr>
<tr>
<td>-1</td>
<td>the last part</td>
</tr>
<tr>
<td>-2</td>
<td>the penultimate part</td>
</tr>
<tr>
<td>2+</td>
<td>the second and all subsequent parts</td>
</tr>
<tr>
<td>-2+</td>
<td>the penultimate and all preceding parts</td>
</tr>
<tr>
<td>1+ and -1+</td>
<td>the same as 0</td>
</tr>
</tbody>
</table>

If N or M is greater than the number of parts available a single underscore is interpolated.
For simple name-based virtual hosts you might use the following directives in your server configuration file:

```
UseCanonicalName Off
VirtualDocumentRoot /usr/local/apache/vhosts/%0
```

A request for http://www.example.com/directory/file.html will be satisfied by the file
/usr/local/apache/vhosts/www.example.com/directory/file.html

For a very large number of virtual hosts it is a good idea to arrange the files to reduce the size of the vhosts directory. To do this you might use the following in your configuration file:

```
UseCanonicalName Off
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

A request for http://www.domain.example.com/directory/file.html will be satisfied by the file
/usr/local/apache/vhosts/example.com/d/o/m/domain/directory/file.html

A more even spread of files can be achieved by hashing from the end of the name, for example:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

The example request would come from
/usr/local/apache/vhosts/example.com/n/i/a/domain.

Alternatively you might use:
VirtualDocumentRoot 
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+

The example request would come from 
/usr/local/apache/vhosts/example.com/d/o/m/a/in/directory/file.html

For IP-based virtual hosting you might use the following in your configuration file:

UseCanonicalName DNS
VirtualDocumentRootIP /usr/local/apache/vhosts/%1/%2/%3/%4/docs
VirtualScriptAliasIP /usr/local/apache/vhosts/%1/%2/%3/%4/cgi-bin

A request for
http://www.domain.example.com/directory/file.html
would be satisfied by the file
/usr/local/apache/vhosts/10/20/30/40/docs/directory/file.html
if the IP address of www.domain.example.com were
10.20.30.40. A request for
http://www.domain.example.com/cgi-bin/script.pl
would be satisfied by executing the program
/usr/local/apache/vhosts/10/20/30/40/cgi-bin/script.pl.

If you want to include the . character in a
VirtualDocumentRoot directive, but it clashes with a %
directive, you can work around the problem in the following way:

VirtualDocumentRoot /usr/local/apache/vhosts/%2.0.%3.0

A request for
http://www.domain.example.com/directory/file.html
will be satisfied by the file
/usr/local/apache/vhosts/domain.example/directory.
The **LogFormat** directives %V and %A are useful in conjunction with this module.
**Description:** Dynamically configure the location of the document root for a given virtual host

**Syntax:** `VirtualDocumentRoot interpolated-directory|none`

**Default:** `VirtualDocumentRoot none`

**Context:** server config, virtual host

**Status:** Extension

**Module:** `mod_vhost_alias`

The `VirtualDocumentRoot` directive allows you to determine where Apache will find your documents based on the value of the server name. The result of expanding `interpolated-directory` is used as the root of the document tree in a similar manner to the `DocumentRoot` directive's argument. If `interpolated-directory` is none then `VirtualDocumentRoot` is turned off. This directive cannot be used in the same context as `VirtualDocumentRootIP`. 
**Description:** Dynamically configure the location of the document root for a given virtual host

**Syntax:** `VirtualDocumentRootIP interpolated-directory|none`

**Default:** `VirtualDocumentRootIP none`

**Context:** server config, virtual host

**Status:** Extension

**Module:** `mod_vhost_alias`

The `VirtualDocumentRootIP` directive is like the `VirtualDocumentRoot` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.
**Description:** Dynamically configure the location of the CGI directory for a given virtual host

**Syntax:**
```
VirtualScriptAlias interpolated-directory|none
```

**Default:** VirtualScriptAlias none

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod_vhost_alias

The `VirtualScriptAlias` directive allows you to determine where Apache will find CGI scripts in a similar manner to `VirtualDocumentRoot` does for other documents. It matches requests for URIs starting `/cgi-bin/`, much like `ScriptAlias /cgi-bin/` would.
| **Description:** | Dynamically configure the location of thecgi
directory for a given virtual host |
| **Syntax:** | `VirtualScriptAliasIP interpolated-directory|none` |
| **Default:** | `VirtualScriptAliasIP none` |
| **Context:** | server config, virtual host |
| **Status:** | Extension |
| **Module:** | `mod_vhost_alias` |

The `VirtualScriptAliasIP` directive is like the
`VirtualScriptAlias` directive, except that it uses the IP
address of the server end of the connection for directory
interpolation instead of the server name.

---

Copyright 2017 The Apache Software Foundation.
Licensed under the [Apache License, Version 2.0](https://www.apache.org/licenses/LICENSE-2.0).

[Modules] [Directives] [FAQ] [Glossary] [Sitemap]
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.
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 ScriptAlias-ed URIs), the type-checking phase (any ScriptAlias-ed request is typed as a CGI script).

The module needs to maintain some per (virtual) server information, namely, the ScriptAlias'es 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 ScriptAlias es 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:

```c
/* Declarations of handlers. */
int translate_scriptalias (request_rec *);
int type_scriptalias (request_rec *);
int cgi_handler (request_rec *);

/* Subsidiary dispatch table for response-phase * handlers, by MIME type */
handler_rec cgi_handlers[] = {
    { "application/x-httpd-cgi", cgi_handler },
    { NULL }
};
```
/* Declarations of routines to manipulate the
 * module's configuration info. Note that these are
 * returned, and passed in, as void *'s; the server
 * core keeps track of them, but it doesn't, and can't,
 * know their internal structure.
 */

void *make_cgi_server_config (pool *);
void *merge_cgi_server_config (pool *, void *, void *);

/* Declarations of routines to handle config-file commands */

extern char *script_alias(cmd_parms *, void *per_dir_config,
char *fake, char *real);

command_rec cgi_cmds[] = {
    { "ScriptAlias", script_alias, NULL, RSRC_CONF, TAKE2,
      "a fakename and a realname"},
    { NULL }
};

module cgi_module = {
    STANDARD_MODULE_STUFF,
    NULL,    /* initializer */
    NULL,    /* dir config creator */
    NULL,    /* dir merger */
    make_cgi_server_config, /* server config */
    merge_cgi_server_config, /* merge server config */
    cgi_cmds,  /* command table */
    cgi_handlers, /* handlers */
    translate_scriptalias, /* filename translation */
    NULL,    /* check_user_id */
    NULL,    /* check auth */
    NULL,    /* check access */
    type_scriptalias, /* type_checker */
    NULL,    /* fixups */
    NULL,    /* logger */
    NULL    /* header parser */
};
The sole argument to handlers is a request_rec structure. This structure describes a particular request which has been made to the server, on behalf of a client. In most cases, each connection to the client generates only one request_rec structure.

A brief tour of the request_rec

The request_rec contains pointers to a resource pool which will be cleared when the server is finished handling the request; to structures containing per-server and per-connection information, and most importantly, information on the request itself.

The most important such information is a small set of character strings describing attributes of the object being requested, including its URI, filename, content-type and content-encoding (these being filled in by the translation and type-check handlers which handle the request, respectively).

Other commonly used data items are tables giving the MIME headers on the client's original request, MIME headers to be sent back with the response (which modules can add to at will), and environment variables for any subprocesses which are spawned off in the course of servicing the request. These tables are manipulated using the ap_table_get and ap_table_set routines.

Note that the Content-type header value cannot be set by module content-handlers using the ap_table_*() routines. Rather, it is set by pointing the content_type field in the request_rec structure to an appropriate string. e.g.,

```
r->content_type = "text/html";
```
Finally, there are pointers to two data structures which, in turn, point to per-module configuration structures. Specifically, these hold pointers to the data structures which the module has built to describe the way it has been configured to operate in a given directory (via .htaccess files or `<Directory>` sections), for private data it has built in the course of servicing the request (so modules' handlers for one phase can pass `notes' to their handlers for other phases). There is another such configuration vector in the `server_rec` data structure pointed to by the `request_rec`, which contains per (virtual) server configuration data.

Here is an abridged declaration, giving the fields most commonly used:

```c
struct request_rec {
    pool *pool;
    conn_rec *connection;
    server_rec *server;

    /* What object is being requested */
    char *uri;
    char *filename;
    char *path_info;
    char *args;  /* QUERY_ARGS, if any */
    struct stat finfo;  /* Set by server core;
                        * st_mode set to zero if no such file */
    char *content_type;
    char *content_encoding;

    /* MIME header environments, in and out. Also,
     * an array containing environment variables to
     * be passed to subprocesses, so people can write
     * modules to add to that environment.
     * 
     * The difference between headers_out and
     * err_headers_out is that the latter are printed
     * even on error, and persist across internal
     * redirects (so the headers printed for
     * `ErrorDocument` handlers will have them).
```
table *headers_in;
table *headers_out;
table *err_headers_out;
table *subprocess_env;

/* Info about the request itself... */

int header_only; /* HEAD request, as opposed to GET */
char *protocol; /* Protocol, as given to us, or HTTP/0.9 */
char *method; /* GET, HEAD, POST, etc. */
int method_number; /* M_GET, M_POST, etc. */
/* Info for logging */
char *the_request;
int bytes_sent;

/* A flag which modules can set, to indicate that
* the data being returned is volatile, and clients
* should be told not to cache it.
*/
int no_cache;

/* Various other config info which may change
* with .htaccess files
* These are config vectors, with one void*
* pointer for each module (the thing pointed
* to being the module's business).
*/
void *per_dir_config; /* Options set in config files, etc. */
void *request_config; /* Notes on *this* request */
};

Where request_rec structures come from
Most request_rec structures are built by reading an HTTP request from a client, and filling in the fields. However, there are a few exceptions:

- If the request is to an imagemap, a type map (i.e., a *.var
file), or a CGI script which returned a local `Location:`, then the resource which the user requested is going to be ultimately located by some URI other than what the client originally supplied. In this case, the server does an *internal redirect*, constructing a new request_rec for the new URI, and processing it almost exactly as if the client had requested the new URI directly.

- If some handler signaled an error, and an ErrorDocument is in scope, the same internal redirect machinery comes into play.
- Finally, a handler occasionally needs to investigate `what would happen if` some other request were run. For instance, the directory indexing module needs to know what MIME type would be assigned to a request for each directory entry, in order to figure out what icon to use.

Such handlers can construct a *sub-request*, using the functions ap_sub_req_lookup_file, ap_sub_req_lookup_uri, and ap_sub_req_method_uri; these construct a new request_rec structure and processes it as you would expect, up to but not including the point of actually sending a response. (These functions skip over the access checks if the sub-request is for a file in the same directory as the original request).

(Server-side includes work by building sub-requests and then actually invoking the response handler for them, via the function ap_run_sub_req).

Handling requests, declining, and returning error codes

As discussed above, each handler, when invoked to handle a
particular request_rec, has to return an int to indicate what happened. That can either be

- OK -- the request was handled successfully. This may or may not terminate the phase.
- DECLINED -- no erroneous condition exists, but the module declines to handle the phase; the server tries to find another.
- an HTTP error code, which aborts handling of the request.

Note that if the error code returned is REDIRECT, then the module should put a Location in the request's headers_out, to indicate where the client should be redirected to.

**Special considerations for response handlers**

Handlers for most phases do their work by simply setting a few fields in the request_rec structure (or, in the case of access checkers, simply by returning the correct error code). However, response handlers have to actually send a request back to the client.

They should begin by sending an HTTP response header, using the function ap_send_http_header. (You don't have to do anything special to skip sending the header for HTTP/0.9 requests; the function figures out on its own that it shouldn't do anything). If the request is marked header_only, that's all they should do; they should return after that, without attempting any further output.

Otherwise, they should produce a request body which responds to the client as appropriate. The primitives for this are ap_rputc and ap_rprintf, for internally generated output, and ap_send_fd, to copy the contents of some FILE * straight to the client.

At this point, you should more or less understand the following
piece of code, which is the handler which handles GET requests which have no more specific handler; it also shows how conditional GETs can be handled, if it's desirable to do so in a particular response handler -- ap_set_last_modified checks against the If-modified-since value supplied by the client, if any, and returns an appropriate code (which will, if nonzero, be USE_LOCAL_COPY). No similar considerations apply for ap_set_content_length, but it returns an error code for symmetry.

```c
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 apRequires.
- Common routines, to handle the protocol end of things, at least for HTTP basic authentication (ap_get_basic_auth_pw, which sets the connection->user structure field automatically, and ap_note_basic_auth_failure, which arranges for the proper WWW-Authenticate: header to be sent back).

**Special considerations for logging handlers**

When a request has internally redirected, there is the question of what to log. Apache handles this by bundling the entire chain of redirects into a list of request_rec structures which are threaded through the r->prev and r->next pointers. The request_rec which is passed to the logging handlers in such cases is the one which was originally built for the initial request from the client; note that the bytes_sent field will only be correct in the last request in
the chain (the one for which a response was actually sent).
One of the problems of writing and designing a server-pool server is that of preventing leakage, that is, allocating resources (memory, open files, etc.), without subsequently releasing them. The resource pool machinery is designed to make it easy to prevent this from happening, by allowing resource to be allocated in such a way that they are automatically released when the server is done with them.

The way this works is as follows: the memory which is allocated, file opened, etc., to deal with a particular request are tied to a resource pool which is allocated for the request. The pool is a data structure which itself tracks the resources in question.

When the request has been processed, the pool is cleared. At that point, all the memory associated with it is released for reuse, all files associated with it are closed, and any other clean-up functions which are associated with the pool are run. When this is over, we can be confident that all the resource tied to the pool have been released, and that none of them have leaked.

Server restarts, and allocation of memory and resources for per-server configuration, are handled in a similar way. There is a configuration pool, which keeps track of resources which were allocated while reading the server configuration files, and handling the commands therein (for instance, the memory that was allocated for per-server module configuration, log files and other files that were opened, and so forth). When the server restarts, and has to reread the configuration files, the configuration pool is cleared, and so the memory and file descriptors which were taken up by reading them the last time are made available for reuse.

It should be noted that use of the pool machinery isn't generally obligatory, except for situations like logging handlers, where you really need to register cleanups to make sure that the log file gets
closed when the server restarts (this is most easily done by using the function `ap_pfopen`, which also arranges for the underlying file descriptor to be closed before any child processes, such as for CGI scripts, are execed), or in case you are using the timeout machinery (which isn't yet even documented here). However, there are two benefits to using it: resources allocated to a pool never leak (even if you allocate a scratch string, and just forget about it); also, for memory allocation, `ap_palalloc` 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_palalloc`, 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:

```c
int my_handler(request_rec *r) {
    struct my_structure *foo;
    ...
    foo = (foo *)ap_palalloc (r->pool, sizeof(my_structure));
}
```

Note that *there is no `ap_pfree`* -- `ap_palalloc`ed memory is freed only when the associated resource pool is cleared. This means that `ap_palalloc` 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_pcaalloc 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:

```c
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 themself, 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->pool1 (especially if r->pool1 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_pfopen, which takes a resource pool and two strings as arguments; the strings are the same as the typical arguments to fopen, *e.g.*,

```c
...
FILE *f = ap_pfopen (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_pfopen, and ap_popenf, namely ap_pfclose and ap_pclosef. (This is because, on many systems, the number of files which a single process can have open is quite limited). It is important to use these functions to close files allocated with ap_pfopen and ap_popenf, since to do otherwise could cause fatal errors on systems such as Linux, which react badly if the same FILE* is closed more than once.

(Using the close functions is not mandatory, since the file will
eventually be closed regardless, but you should consider it in cases where your module is opening, or could open, a lot of files).

**Other sorts of resources -- cleanup functions**

More text goes here. Describe the cleanup primitives in terms of which the file stuff is implemented; also, spawn_process.

Pool cleanups live until `clear_pool()` is called:

`clear_pool(a)` recursively calls `destroy_pool()` on all subpools of `a`; then calls all the cleanups for `a`; then releases all the memory for `a`. `destroy_pool(a)` calls `clear_pool(a)` and then releases the pool structure itself. *i.e.*, `clear_pool(a)` doesn't delete `a`, it just frees up all the resources and you can start using it again immediately.

**Fine control -- creating and dealing with sub-pools, with a note on sub-requests**

On rare occasions, too-free use of `ap_palloc()` and the associated primitives may result in undesirably profligate resource allocation. You can deal with such a case by creating a sub-pool, allocating within the sub-pool rather than the main pool, and clearing or destroying the sub-pool, which releases the resources which were associated with it. (This really is a rare situation; the only case in which it comes up in the standard module set is in case of listing directories, and then only with very large directories. Unnecessary use of the primitives discussed here can hair up your code quite a bit, with very little gain).

The primitive for creating a sub-pool is `ap_make_sub_pool`, which takes another pool (the parent pool) as an argument. When the main pool is cleared, the sub-pool will be destroyed. The sub-pool may also be cleared or destroyed at any time, by calling the functions `ap_clear_pool` and `ap_destroy_pool`, respectively.
(The difference is that ap_clear_pool frees resources associated with the pool, while ap_destroy_pool also deallocates the pool itself. In the former case, you can allocate new resources within the pool, and clear it again, and so forth; in the latter case, it is simply gone).

One final note -- sub-requests have their own resource pools, which are sub-pools of the resource pool for the main request. The polite way to reclaim the resources associated with a sub request which you have allocated (using the ap_sub_req... functions) is ap_destroy_sub_req, which frees the resource pool. Before calling this function, be sure to copy anything that you care about which might be allocated in the sub-request's resource pool into someplace a little less volatile (for instance, the filename in its request_rec structure).

(Again, under most circumstances, you shouldn't feel obliged to call this function; only 2K of memory or so are allocated for a typical sub request, and it will be freed anyway when the main request pool is cleared. It is only when you are allocating many, many sub-requests for a single main request that you should seriously consider the ap_destroy... functions).
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 srm.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:

```c
typedef struct {
    table *forced_types; /* Additional AddTyped stuff */
    table *encoding_types; /* Added with AddEncoding... */
} mime_dir_config;
```

When the server is reading a configuration file, or `<Directory>` section, which includes one of the MIME module's commands, it needs to create a `mime_dir_config` structure, so those commands have something to act on. It does this by invoking the function it finds in the module's `create per-dir config slot`, with two arguments: the name of the directory to which this configuration
information applies (or NULL for srm.conf), and a pointer to a resource pool in which the allocation should happen.

(If we are reading a .htaccess file, that resource pool is the per-request resource pool for the request; otherwise it is a resource pool which is used for configuration data, and cleared on restarts. Either way, it is important for the structure being created to vanish when the pool is cleared, by registering a cleanup on the pool if necessary).

For the MIME module, the per-dir config creation function just ap_pallocs the structure above, and a creates a couple of tables to fill it. That looks like this:

```c
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:

```c
void *merge_mime_dir_configs (pool *p, void *parent_dirv, void *subdirv)
{
    mime_dir_config *parent_dir = (mime_dir_config *)parent_dirv;
    mime_dir_config *subdir = (mime_dir_config *)subdirv;
    mime_dir_config *new = (mime_dir_config *)ap_palloc (p, sizeof (mime_dir_config));

    new->forced_types = ap_overlay_tables (p, subdir->forced_types,
        parent_dir->forced_types);
    new->encoding_types = ap_overlay_tables (p, subdir->encoding_types,
        parent_dir->encoding_types);

    return new;
}
```

As a note -- if there is no per-directory merge function present, the server will just use the subdirectory's configuration info, and ignore the parent's. For some modules, that works just fine (e.g., for the includes module, whose per-directory configuration information consists solely of the state of the XBITHACK), and for those modules, you can just not declare one, and leave the corresponding structure slot in the module itself NULL.

**Command handling**

Now that we have these structures, we need to be able to figure out how to fill them. That involves processing the actual AddType and AddEncoding commands. To find commands, the server looks in the module's command table. That table contains information on how many arguments the commands take, and in what formats, where it is permitted, and so forth. That information is sufficient to allow the server to invoke most command-handling functions with pre-parsed arguments. Without further ado, let's
look at the **AddType** command handler, which looks like this (the **AddEncoding** command looks basically the same, and won't be shown here):

```c
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:
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.

```c
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):

```c
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;
}
```
Debugging Memory Allocation in APR

This document has been removed.
Documenting Apache 2.0

Apache 2.0 uses Doxygen to document the APIs and global variables in the code. This will explain the basics of how to document using Doxygen.
To start a documentation block, use /**
To end a documentation block, use */

In the middle of the block, there are multiple tags we can use:

- **Description of this function's purpose**
  - @param parameter_name description
  - @return description
  - @deffunc signature of the function

The deffunc is not always necessary. DoxyGen does not have a full parser in it, so any prototype that use a macro in the return type declaration is too complex for scandoc. Those functions require a deffunc. An example (using &gt; rather than >):

```c
/**
 * 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:

```c
/**
 * @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.
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.
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:

```c
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:

```c
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:

```c
AP_IMPLEMENT_HOOK_RUN_FIRST(int, do_something, (request_rec *r, int n), (r, n), DECLINED)
```

The first hook that does *not* return DECLINED stops the loop and its return value is returned from the hook caller. Note that DECLINED is the traditional hook return value meaning "I didn't do anything", but it can be whatever suits you.

Alternatively, all hooks can be run until an error occurs. This boils down to permitting two return values, one of which means "I did something, and it was OK" and the other meaning "I did nothing". The first function that returns a value other than one of those two stops the loop, and its return is the return value. Declare these like so:

```c
AP_IMPLEMENT_HOOK_RUN_ALL(int, do_something, (request_rec *r,`
Again, OK and DECLINED are the traditional values. You can use what you want.

**Call the hook callers**

At appropriate moments in the code, call the hook caller, like so:

```c
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:

```c
static int my_something_doer(request_rec *r, int n)
{
    ...
    return OK;
}
```

**Add a hook registering function**

During initialisation, the server will call each modules hook registering function, which is included in the module structure:

```c
static void my_register_hooks()
{
    ap_hook_do_something(my_something_doer, NULL, NULL,
                          APR_HOOK_MIDDLE);
}

mode MODULE_VAR_EXPORT my_module =
{
    ...
    my_register_hooks /* register hooks */
};
```

**Controlling hook calling order**

In the example above, we didn't use the three arguments in the hook registration function that control calling order. There are two mechanisms for doing this. The first, rather crude, method, allows us to specify roughly where the hook is run relative to other modules. The final argument control this. There are three possible values: APR_HOOK_FIRST, APR_HOOK_MIDDLE and
APR_HOOK_LAST.

All modules using any particular value may be run in any order relative to each other, but, of course, all modules using APR_HOOK_FIRST will be run before APR_HOOK_MIDDLE which are before APR_HOOK_LAST. Modules that don't care when they are run should use APR_HOOK_MIDDLE. These values are spaced out, so that positions like APR_HOOK_FIRST-2 are possible to hook slightly earlier than other functions.

Note that there are two more values, APR_HOOK_REALLY_FIRST and APR_HOOK_REALLY_LAST. These should only be used by the hook exporter.

The other method allows finer control. When a module knows that it must be run before (or after) some other modules, it can specify them by name. The second (third) argument is a NULL-terminated array of strings consisting of the names of modules that must be run before (after) the current module. For example, suppose we want "mod_xyz.c" and "mod_abc.c" to run before we do, then we'd hook as follows:

```c
static void register_hooks()
{
    static const char * constaszPre[] = { "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 1.3  Apache 2.0

mod_mmap_static  Apache 2.0
apr_status_t apr_status_t
ARP_SUCCESS

- apr_pool_t *p
- apr_pool_t *plog
- apr_pool_t *ptemp
- server_rec *s

APR

- pool becomes apr_pool_t
- table becomes apr_table_t
mod_mmap_static:

```c
static void register_hooks(void)
{
    static const char * const aszPre[]={ "http_core.c", NULL };
    ap_hook_post_config(mmap_post_config, NULL, NULL, HOOK_MIDDLE);
    ap_hook_translate_name(mmap_static_xlat, aszPre, NULL, HOOK_LAST);
}
```

post_config(

```c
    ap_hook_phase_name(function_name, predecessors, successors, position);
```

...

- HOOK_FIRST
- HOOK_MIDDLE
- HOOK_LAST

```c
mod_mmap_static  post_config
mmap_static_xlat  core
```

```c
aszPre
```
module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD_MODULE_STUFF,
    / initializer */
    / dir config creator */
    / dir merger --- default is to override */
    / server config */
    / merge server config */
    / command handlers */
    / handlers */
    / filename translation */
    / check_user_id */
    / check auth */
    / check access */
    / type_checker */
    / fixups */
    / logger */
    / header parser */
    / child_init */
    / child_exit */
    / post read-request */
};

...
ap_hook_post_config
  (_init)
ap_hook_http_method
  (HTTP ())
ap_hook_open_logs
    ()
ap_hook_auth_checker
    ()
ap_hook_access_checker
    ()
ap_hook_check_user_id
    (ID )
ap_hook_default_port
    ()
ap_hook_pre_connection
    ( accept)
ap_hook_process_connection
    ()
ap_hook_child_init
   ()
ap_hook_create_request
   (??)
ap_hook_fixups
   ()
ap_hook_handler
   ()
ap_hook_header_parser
   (post_read_request )
ap_hook_insert_filter
   ()
ap_hook_log_transaction
   ()
ap_hook_optional_fn_retrieve
   ()
ap_hook_post_read_request
   ()
ap_hook_quick_handler

ap_hook_translate_name
   (URI )
ap_hook_type_checker
   ()
Several changes in 2.0 and above affect the internal request processing mechanics. Module authors need to be aware of these changes so they may take advantage of the optimizations and security enhancements.

The first major change is to the subrequest and redirect mechanisms. There were a number of different code paths in the Apache HTTP Server 1.3 to attempt to optimize subrequest or redirect behavior. As patches were introduced to 2.0, these optimizations (and the server behavior) were quickly broken due to this duplication of code. All duplicate code has been folded back into `ap_process_request_internal()` to prevent the code from falling out of sync again.

This means that much of the existing code was 'unoptimized'. It is the Apache HTTP Project's first goal to create a robust and correct implementation of the HTTP server RFC. Additional goals include security, scalability and optimization. New methods were sought to optimize the server (beyond the performance of 1.3) without introducing fragile or insecure code.
All requests pass through `ap_process_request_internal()` in `request.c`, including subrequests and redirects. If a module doesn't pass generated requests through this code, the author is cautioned that the module may be broken by future changes to request processing.

To streamline requests, the module author can take advantage of the hooks offered to drop out of the request cycle early, or to bypass core hooks which are irrelevant (and costly in terms of CPU.)
Unescapes the URL

The request's parsed_uri path is unescaped, once and only once, at the beginning of internal request processing.

This step is bypassed if the proxyreq flag is set, or the parsed_uri.path element is unset. The module has no further control of this one-time unescape operation, either failing to unescape or multiply unescaping the URL leads to security repercussions.

Strips Parent and This Elements from the URI

All ../../ and ../../../ elements are removed by ap_getparents(). This helps to ensure the path is (nearly) absolute before the request processing continues.

This step cannot be bypassed.

Initial URI Location Walk

Every request is subject to an ap_location_walk() call. This ensures that <Location> sections are consistently enforced for all requests. If the request is an internal redirect or a sub-request, it may borrow some or all of the processing from the previous or parent request's ap_location_walk, so this step is generally very efficient after processing the main request.

translate_name

Modules can determine the file name, or alter the given URI in this step. For example, mod_vhost_alias will translate the URI's path into the configured virtual host, mod_alias will translate the path to an alias path, and if the request falls back on the core, the
**DocumentRoot** is prepended to the request resource.

If all modules DECLINE this phase, an error 500 is returned to the browser, and a "couldn't translate name" error is logged automatically.

**Hook: map_to_storage**

After the file or correct URI was determined, the appropriate per-dir configurations are merged together. For example, `mod_proxy` compares and merges the appropriate `<Proxy>` sections. If the URI is nothing more than a local (non-proxy) TRACE request, the core handles the request and returns DONE. If no module answers this hook with OK or DONE, the core will run the request filename against the `<Directory>` and `<Files>` sections. If the request 'filename' isn't an absolute, legal filename, a note is set for later termination.

**URI Location Walk**

Every request is hardened by a second `ap_location_walk()` call. This reassures that a translated request is still subjected to the configured `<Location>` sections. The request again borrows some or all of the processing from its previous `location_walk` above, so this step is almost always very efficient unless the translated URI mapped to a substantially different path or Virtual Host.

**Hook: header_parser**

The main request then parses the client's headers. This prepares the remaining request processing steps to better serve the client's request.
switch (ap_satisfies(r)) {
    case SATISFY_ALL:
    case SATISFY_NOSPEC:
        if ((access_status = ap_run_access_checker(r)) != 0) {
            return decl_die(access_status, "check access", r);
        }

        if (ap_some_auth_required(r)) {
            if (((access_status = ap_run_check_user_id(r)) != 0) || !ap_auth_type(r)) {
                return decl_die(access_status, ap_auth_type(r) ? "check user. No user file?" : "perform authentication. AuthType not set!", r);
            }
        }

        if (((access_status = ap_run_auth_checker(r)) != 0) || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r) ? "check access. No groups file?" : "perform authentication. AuthType not set!", r);
        }
    }
    break;
}

case SATISFY_ANY:
    if (((access_status = ap_run_access_checker(r)) != 0)) {
        if (!ap_some_auth_required(r)) {
            return decl_die(access_status, "check access", r);
        }
    }

    if (((access_status = ap_run_check_user_id(r)) != 0) || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r) ? "check user. No user file?" : "perform authentication. AuthType not set!", r);
    }

    if (((access_status = ap_run_auth_checker(r)) != 0) || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r) ? "check access. No groups file?" : "perform authentication. AuthType not set!", r);
    }
r);

} } break;
}
The Preparation Phase

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.
Warning
This is a cut 'n paste job from an email (<022501c1c529$f63a9550$7f00000a@KOJ>) and only reformatted for better readability. It's not up to date but may be a good start for further research.
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_FTYPERESOURCE, AP_FTYPECONTENT_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 were 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
Apache

APache eXtension Tool (apxs)

Apache Web perl

Apache Portable Runtime (APR)

APache HTTP Server OS

Apache HTTP Server

Apache Portable Runtime

(subject)

CA

SSL/TLS

Certificate Signing Request (CSR)

CA

CSR

SSL/TLS

Certification Authority (CA)

SSL/TLS

Cipher

DES, IDEA, RC4

SSL/TLS
Cipher: SSL/TLS

Common Gateway Interface (CGI)

: CGI

Apache

CONNECT
  HTTP HTTP SSL

: Apache httpd

( Dynamic Shared Object) (DSO)

Apache

(env-variable)

: Apache

(: Export-Crippled)

Export Administration Regulations (EAR)
SSL/TLS

(Full-Qualified Domain-Name) (FQDN)
IP
www.example.com

Apache

CGI

Apache

httpd.conf

httpd.conf

World Wide Web Apache

HyperText Transfer Protocol (HTTP)

HTTP/1.1

HTTPS

The HyperText Transfer Protocol (Secure), World Wide Web

SSL HTTP
HTTP

: SSL/TLS

MIME

Multipurpose Internet Mail Extensions (MIME)
text/html, image/gif, application/octet-stream

HTTP MIME

: mod_mime

Apache

Apache HTTP

: tarball

Apache

Module Magic Number (MMN)

Apache

Apache

OpenSSL

SSL/TLS

http://www.openssl.org/

: SSL/TLS

: SSL/TLS

: mod_proxy
SSL/TLS

(Regex)

Apache
/images/.*\(jpg|gif\)\$

(: Perl Compatible Regular Expressions)

Secure Sockets Layer (SSL)
Netscape Communications Corporation TCP/IP
HTTPS SSL HyperText Transfer Protocol (HTTP)

Server Side Includes (SSI)
HTML

SSL/TLS

SSLeay
Eric A. Young SSL/TLS

Cipher

Tarball
tar

Transport Layer Security (TLS)
TCP/IP Internet Engineering
TLS 1 SSL 3
: SSL/TLS

Uniform Resource Locator (URL)
Internet
http  https  URL
http://httpd.apache.org/docs/2.2/glossary.html

Uniform Resource Identifier (URI)

Apache  IP  IP
: Apache

X.509
SSL/TLS  International  Telecommunication Union
: SSL/TLS

Copyright 2017 The Apache Software Foundation.
Apache License, Version 2.0
Apache

- 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_No UserID
- Anonymous_VerifyEmail
- AuthBasicAuthoritative
- AuthBasicProvider
- AuthDBDUserPWQuery
- AuthDBDUserRealmQuery
- AuthDBMGroupFile
- AuthDBMType
- AuthDBMUserFile
- AuthDefaultAuthoritative
- AuthDigestAlgorithm
- AuthDigestDomain
- AuthDigestNcCheck
- AuthDigestNonceFormat
- AuthDigestNonceLifetime
- AuthDigestProvider
- AuthDigestQop
- AuthDigestShmemSize
- AuthGroupFile
- AuthLDAPBindAuthoritative
- AuthLDAPBindDN
- AuthLDAPBindPassword
- AuthLDAPCharsetConfig
- AuthLDAPCompareDNOnServer
- AuthLDAPDereferenceAliases
- AuthLDAPGroupAttribute
- AuthLDAPGroupAttributeIsDN
- AuthLDAPRemoteUserAttribute
- AuthLDAPRemoteUserIsDN
- AuthLDAPUrl
- AuthName
- <AuthnProviderAlias>
- AuthType
- AuthUserFile
- AuthzDBMAuthoritative
- AuthzDBMType
- AuthzDefaultAuthoritative
- AuthzGroupFileAuthoritative
- AuthzLDAPAuthoritative
- AuthzOwnerAuthoritative
- AuthzUserAuthoritative
- BalancerMember
- BrowserMatch
- BrowserMatchNoCase
- BufferedLogs
- CacheDefaultExpire
- CacheDirLength
- CacheDirLevels
- CacheDisable
- CacheEnable
- CacheFile
- CacheIgnoreCacheControl
- CacheIgnoreHeaders
- CacheIgnoreNoLastMod
- CacheIgnoreQueryString
- CacheIgnoreURLSessionIdentifiers
- CacheLastModifiedFactor
- CacheLock
- CacheLockMaxAge
- CacheLockPath
- CacheMaxExpire
- CacheMaxFileSize
- CacheMinFileSize
- CacheNegotiatedDocs
- CacheRoot
- CacheStoreNoStore
- CacheStorePrivate
- CGIDScriptTimeout
- CGIMapExtension
- CharSetDefault
- CharSetOptions
- CharSetSourceEnc
- CheckCaseOnly
- CheckSpelling
- ChrootDir
- ContentDigest
- CookieDomain
- CookieExpires
- CookieLog
- CookieName
- CookieStyle
- CookieTracking
- CoreDumpDirectory
- CustomLog
- Dav
- DavDepthInfinity
- DavGenericLockDB
- DavLockDB
- DavMinTimeout
- DBDExptime
- DBDKeep
- DBDMax
- DBDMin
- DBDParams
- DBDPersist
- DBDPrepareSQL
• **DBDriver**
• **DefaultIcon**
• **DefaultLanguage**
• **DefaultType**
• **DeflateBufferSize**
• **DeflateCompressionLevel**
• **DeflateFilterNote**
• **DeflateInflateLimitRequestBody**
• **DeflateInflateRatioBurst**
• **DeflateInflateRatioLimit**
• **DeflateMemLevel**
• **DeflateWindowSize**
• **Deny**
• **<Directory>**
• **DirectoryIndex**
• **<DirectoryMatch>**
• **DirectorySlash**
• **DocumentRoot**
• **DumpIOInput**
• **DumpIOLogLevel**
• **DumpIOOutput**
• **EnableExceptionHook**
• **EnableMMAP**
• **EnableSendfile**
• **ErrorDocument**
• **ErrorLog**
• **Example**
• **ExpiresActive**
• **ExpiresByType**
• **ExpiresDefault**
• **ExtendedStatus**
• **ExtFilterDefine**
• **ExtFilterOptions**
• **FallbackResource**
- FileETag
- <Files>
- <FilesMatch>
- FilterChain
- FilterDeclare
- FilterProtocol
- FilterProvider
- FilterTrace
- ForceLanguagePriority
- ForceType
- ForensicLog
- GprofDir
- GracefulShutdownTimeout
- Group
- Header
- HeaderName
- HostnameLookups
- HttpProtocolOptions
- IdentityCheck
- IdentityCheckTimeout
- <IfDefine>
- <IfModule>
- <IfVersion>
- ImapBase
- ImapDefault
- ImapMenu
- Include
- IndexHeadInsert
- IndexIgnore
- IndexOptions
- IndexOrderDefault
- IndexStyleSheet
- ISAPIAppendLogToErrors
- ISAPIAppendLogToQuery
- ISAPICacheFile
- ISAPIFakeAsync
- ISAPILogNotSupported
- ISAPIReadAheadBuffer
- KeepAlive
- KeepAliveTimeout
- LanguagePriority
- LDAPCacheEntries
- LDAPCacheTTL
- LDAPConnectionTimeout
- LDAPOpCacheEntries
- LDAPOpCacheTTL
- LDAPSharedCacheFile
- LDAPSharedCacheSize
- LDAPTrustedClientCert
- LDAPTrustedGlobalCert
- LDAPTrustedMode
- LDAPVerifyServerCert
- <Limit>
- <LimitExcept>
- LimitInternalRecursion
- LimitRequestBody
- LimitRequestFields
- LimitRequestFieldSize
- LimitRequestLine
- LimitXMLRequestBody
- Listen
- ListenBackLog
- LoadFile
- LoadModule
- <Location>
- <LocationMatch>
- LockFile
- LogFormat
- LogLevel
- MaxClients
- MaxKeepAliveRequests
- MaxMemFree
- MaxRanges
- MaxRequestsPerChild
- MaxRequestsPerThread
- MaxSpareServers
- MaxSpareThreads
- MaxThreads
- MCacheMaxObjectCount
- MCacheMaxObjectSize
- MCacheMaxStreamingBuffer
- MCacheMinObjectSize
- MCacheRemovalAlgorithm
- MCacheSize
- MergeTrailers
- MetaDir
- MetaFiles
- MetaSuffix
- MimeMagicFile
- MinSpareServers
- MinSpareThreads
- MMapFile
- ModMimeUsePathInfo
- MultiviewsMatch
- NameVirtualHost
- NoProxy
- NWSSLTrustedCerts
- NWSSLUrgeable
- Options
- Order
- PassEnv
- PidFile
- Protocol
- ProtocolEcho
- <Proxy>
- ProxyBadHeader
- ProxyBlock
- ProxyDomain
- ProxyErrorOverride
- ProxyFtpDirCharset
- ProxyIObufferSize
- <ProxyMatch>
- ProxyMaxForwards
- ProxyPass
- ProxyPassInterpolateEnv
- ProxyPassMatch
- ProxyPassReverse
- ProxyPassReverseCookieDomain
- ProxyPassReverseCookiePath
- ProxyPreserveHost
- ProxyReceiveBufferSize
- ProxyRemote
- ProxyRemoteMatch
- ProxyRequests
- ProxySCGIPInternalRedirect
- ProxySCGISendfile
- ProxySet
- ProxyStatus
- ProxyTimeout
- ProxyVia
- ReadmeName
- ReceiveBufferSize
- Redirect
- RedirectMatch
- RedirectPermanent
- RedirectTemp
- RegisterHttpMethod
- RemoveCharset
- RemoveEncoding
- RemoveHandler
- RemoveInputFilter
- RemoveLanguage
- RemoveOutputFilter
- RemoveType
- RequestHeader
- RequestReadTimeout
- Require
- RewriteBase
- RewriteCond
- RewriteEngine
- RewriteLock
- RewriteLog
- RewriteLogLevel
- RewriteMap
- RewriteOptions
- RewriteRule
- RLimitCPU
- RLimitMEM
- RLimitNPROC
- Satisfy
- ScoreBoardFile
- Script
- ScriptAlias
- ScriptAliasMatch
- ScriptInterpreterSource
- ScriptLog
- ScriptLogBuffer
- ScriptLogLength
- ScriptSock
- SecureListen
- SeeRequestTail
- SendBufferSize
- ServerAdmin
- ServerAlias
- ServerLimit
- ServerName
- ServerPath
- ServerRoot
- ServerSignature
- ServerTokens
- SetEnv
- SetEnvIf
- SetEnvIfNoCase
- SetHandler
- SetInputFilter
- SetOutputFilter
- SSIEnableAccess
- SSIEndTag
- SSIEErrorMsg
- SSIETag
- SSLLastModified
- SSISTartTag
- SSITimeFormat
- SSIUndefinedEcho
- SSLCACertificateFile
- SSLCACertificatePath
- SSLCADNRequestFile
- SSLCADNRequestPath
- SSLCAREvocationFile
- SSLCAREvocationPath
- SSLCertificateChainFile
- SSLCertificateFile
- SSLCertificateKeyFile
- SSLCipherSuite
- SSLCompression
- SSLCryptoDevice
- SSLEngine
- SSLFIPS
- SSLHonorCipherOrder
- SSLInsecureRenegotiation
- SSLMutex
- SSLOptions
- SSLPassPhraseDialog
- SSLProtocol
- SSLProxyCACertificateFile
- SSLProxyCACertificatePath
- SSLProxyCARevocationFile
- SSLProxyCARevocationPath
- SSLProxyCheckPeerCN
- SSLProxyCheckPeerExpire
- SSLProxyCipherSuite
- SSLProxyEngine
- SSLProxyMachineCertificateChainFile
- SSLProxyMachineCertificateFile
- SSLProxyMachineCertificatePath
- SSLProxyProtocol
- SSLProxyVerify
- SSLProxyVerifyDepth
- SSLRandomSeed
- SSLRenegBufferSize
- SSLRequire
- SSLRequireSSL
- SSLSessionCache
- SSLSessionCacheTimeout
- SSLSessionTicketKeyFile
- SSLSessionTickets
- SSLStrictSNIVHostCheck
- SSLUserName
SSLVerifyClient
SSLVerifyDepth
StartServers
StartThreads
Substitute
SubstituteInheritBefore
Suexec
SuexecUserGroup
ThreadLimit
ThreadsPerChild
ThreadStackSize
TimeOut
TraceEnable
TransferLog
TypesConfig
UnsetEnv
UseCanonicalName
UseCanonicalPhysicalPort
User
UserDir
VirtualDocumentRoot
VirtualDocumentRootIP
<VirtualHost>
VirtualScriptAlias
VirtualScriptAliasIP
Win32DisableAcceptEx
XBitHack
### Apache

<table>
<thead>
<tr>
<th>AcceptFilter protocol accept_filter</th>
<th>Listen</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcceptMutex default</td>
<td>method</td>
</tr>
<tr>
<td>AcceptPathInfo On</td>
<td>Off</td>
</tr>
<tr>
<td>AccessFileName filename [filename] ...</td>
<td>.htaccess</td>
</tr>
<tr>
<td>Action action-type cgi-script [virtual]</td>
<td>CGI</td>
</tr>
<tr>
<td>AddAlt string file [file] ...</td>
<td></td>
</tr>
<tr>
<td>AddAltByEncoding string MIME-encoding [MIME-encoding] ...</td>
<td>MIME</td>
</tr>
<tr>
<td>AddAltByType string MIME-type [MIME-type] ...</td>
<td>MPEG</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>AddCharset <strong>charset</strong></td>
<td>Configures the character set for a file.</td>
</tr>
<tr>
<td>AddDefaultCharset <strong>On</strong></td>
<td>Enables or disables default character set.</td>
</tr>
<tr>
<td></td>
<td>text/plain text/html charset</td>
</tr>
<tr>
<td></td>
<td><strong>Off</strong></td>
</tr>
<tr>
<td>AddDescription <strong>string</strong></td>
<td>Associates a description with a file.</td>
</tr>
<tr>
<td>AddEncoding <strong>MIME-enc</strong></td>
<td>Adds an encoding to a file.</td>
</tr>
<tr>
<td>AddHandler <strong>handler-name</strong></td>
<td>Associates a handler with an extension.</td>
</tr>
<tr>
<td>AddIcon <strong>icon name</strong></td>
<td>Associates an icon with a file.</td>
</tr>
<tr>
<td>AddIconByEncoding <strong>icon</strong></td>
<td>Associates an icon with a MIME-encoding.</td>
</tr>
<tr>
<td>AddIconByType <strong>icon</strong></td>
<td>Associates an icon with a MIME-type.</td>
</tr>
<tr>
<td>AddInputFilter <strong>filter</strong></td>
<td>Associates an input filter with an extension.</td>
</tr>
<tr>
<td>AddLanguage <strong>MIME-lang</strong></td>
<td>Associates a language with an extension.</td>
</tr>
<tr>
<td>AddModuleInfo <strong>module-name</strong></td>
<td>Associates a module with a string.</td>
</tr>
<tr>
<td>AddOutputFilter <strong>filter</strong></td>
<td>Associates an output filter with an extension.</td>
</tr>
<tr>
<td>AddOutputFilterByType <strong>filter</strong></td>
<td>Associates an output filter by MIME-type.</td>
</tr>
<tr>
<td>AddType <strong>MIME-type</strong></td>
<td>Associates a MIME type with an extension.</td>
</tr>
<tr>
<td>alias</td>
<td>URL-path file-path</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>alias</td>
<td>URL</td>
</tr>
<tr>
<td>aliasmatch</td>
<td>regex file-path</td>
</tr>
<tr>
<td>allow from all</td>
<td>host env=env-variable</td>
</tr>
<tr>
<td>allow connect</td>
<td>port [port]</td>
</tr>
<tr>
<td>allow encoded slashes</td>
<td>on/off</td>
</tr>
<tr>
<td>allow override</td>
<td>all none</td>
</tr>
<tr>
<td>allow override</td>
<td>all none</td>
</tr>
<tr>
<td>anonymous</td>
<td>user [user]...</td>
</tr>
<tr>
<td>anonymous log email</td>
<td>on/off</td>
</tr>
<tr>
<td>anonymous must give email</td>
<td>on/off</td>
</tr>
<tr>
<td>anonymous no user id</td>
<td>on/off</td>
</tr>
<tr>
<td>anonymous verify email</td>
<td>on/off</td>
</tr>
<tr>
<td>auth basic authoritative</td>
<td>on/off</td>
</tr>
<tr>
<td>auth basic provider</td>
<td>on/off provider-name</td>
</tr>
<tr>
<td>auth dbd user pw query</td>
<td>sql query</td>
</tr>
<tr>
<td>auth dbd user realm query</td>
<td>sql query</td>
</tr>
<tr>
<td>auth dbm group file</td>
<td>file-path</td>
</tr>
<tr>
<td><strong>Sets the name of the database file containing the list of user groups for authorization</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>AuthDBMType</strong></td>
<td>default</td>
</tr>
<tr>
<td><strong>AuthDBMUserFile</strong></td>
<td>file-path</td>
</tr>
<tr>
<td><strong>AuthDefaultAuthoritative</strong></td>
<td>On</td>
</tr>
<tr>
<td><strong>AuthDigestAlgorithm</strong></td>
<td>MD5</td>
</tr>
<tr>
<td><strong>AuthDigestDomain</strong></td>
<td>URI [URI]...</td>
</tr>
<tr>
<td><strong>AuthDigestNonceFormat</strong></td>
<td>format</td>
</tr>
<tr>
<td><strong>AuthDigestNonceLifetime</strong></td>
<td>seconds</td>
</tr>
<tr>
<td><strong>AuthDigestProvider</strong></td>
<td>provider-name [provider-name]...</td>
</tr>
<tr>
<td><strong>AuthDigestQop</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>AuthDigestShmemSize</strong></td>
<td>size</td>
</tr>
<tr>
<td><strong>AuthGroupFile</strong></td>
<td>file-path</td>
</tr>
<tr>
<td><strong>AuthLDAPBindAuthoritative</strong></td>
<td>off</td>
</tr>
<tr>
<td><strong>AuthLDAPBindDN</strong></td>
<td>distinguished-name</td>
</tr>
<tr>
<td><strong>AuthLDAPBindPassword</strong></td>
<td>password</td>
</tr>
</tbody>
</table>

- **AuthDBMType**: default | SDBM|GDBM|NDBM|DB
- **AuthDBMUserFile**: file-path
- **AuthDefaultAuthoritative**: On
- **AuthDigestAlgorithm**: MD5
- **AuthDigestDomain**: URI [URI]...
- **AuthDigestNonceFormat**: format
- **AuthDigestNonceLifetime**: 300 seconds
- **AuthDigestProvider**: provider-name [provider-name]...
- **AuthDigestQop**: auth
- **AuthDigestShmemSize**: 1000 size
- **AuthGroupFile**: file-path
- **AuthLDAPBindAuthoritative**: on
- **AuthLDAPBindDN**: distinguished-name
- **AuthLDAPBindPassword**: password
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthLDAPCharsetConfig <code>file-path</code></td>
<td>Language to charset conversion configuration file</td>
</tr>
<tr>
<td>AuthLDAPCompareDNOnServer `on</td>
<td>off`</td>
</tr>
<tr>
<td>Use the LDAP server to compare the DNs</td>
<td></td>
</tr>
<tr>
<td>AuthLDAPDereferenceAliases `never</td>
<td>searching</td>
</tr>
<tr>
<td>When will the module de-reference aliases</td>
<td></td>
</tr>
<tr>
<td>AuthLDAPGroupAttribute <code>attribute</code></td>
<td><code>member uniquememb +</code></td>
</tr>
<tr>
<td>LDAP attributes used to check for group membership</td>
<td></td>
</tr>
<tr>
<td>AuthLDAPGroupAttributeIsDN `on</td>
<td>off`</td>
</tr>
<tr>
<td>Use the DN of the client username when checking for group membership</td>
<td></td>
</tr>
<tr>
<td>AuthLDAPRemoteUserAttribute <code>uid</code></td>
<td>Use the value of the attribute returned during the user query to set the REMOTE_USER environment variable</td>
</tr>
<tr>
<td>AuthLDAPRemoteUserIsDN `on</td>
<td>off`</td>
</tr>
<tr>
<td>Use the DN of the client username to set the REMOTE_USER environment variable</td>
<td></td>
</tr>
<tr>
<td>AuthLDAPUrl `url [NONE</td>
<td>SSL</td>
</tr>
<tr>
<td>AuthName <code>auth-domain</code></td>
<td>HTTP (: realm)</td>
</tr>
<tr>
<td><code>&lt;AuthnProviderAlias baseProvider Alias&gt; ...</code></td>
<td>Enclose a group of directives that represent an extension of a base authentication provider by the specified alias</td>
</tr>
<tr>
<td>AuthType Basic</td>
<td>Digest</td>
</tr>
<tr>
<td>AuthUserFile <code>file-path</code></td>
<td></td>
</tr>
<tr>
<td>AuthzDBMAuthoritative `On</td>
<td>Off`</td>
</tr>
<tr>
<td>Sets whether authorization will be passed on to lower level modules</td>
<td></td>
</tr>
<tr>
<td>AuthzDBMType `default</td>
<td>SDBM</td>
</tr>
<tr>
<td>Sets the type of database file that is used to store list of user groups</td>
<td></td>
</tr>
<tr>
<td>AuthzDefaultAuthoritative `On</td>
<td>Off`</td>
</tr>
<tr>
<td>Configuration Setting</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>AuthzGroupFileAuthoritative</td>
<td>On</td>
</tr>
<tr>
<td>AuthzLDAPAuthoritative</td>
<td>On</td>
</tr>
<tr>
<td>Prevent other authentication modules from authenticating the user if this one fails</td>
<td></td>
</tr>
<tr>
<td>AuthzOwnerAuthoritative</td>
<td>On</td>
</tr>
<tr>
<td>AuthzUserAuthoritative</td>
<td>On</td>
</tr>
<tr>
<td>BalancerMember</td>
<td></td>
</tr>
<tr>
<td>[balancerurl] url [key=value [key=value ...]]</td>
<td></td>
</tr>
<tr>
<td>BrowserMatch</td>
<td></td>
</tr>
<tr>
<td>regex [!]env-variable[=value] [!]env-variable[=value] ...</td>
<td>HTTP User-Agent</td>
</tr>
<tr>
<td>BrowserMatchNoCase</td>
<td></td>
</tr>
<tr>
<td>regex [!]env-variable[=value] [!]env-variable[=value] ...</td>
<td>HTTP User-Agent</td>
</tr>
<tr>
<td>BufferedLogs</td>
<td>Off</td>
</tr>
<tr>
<td>CacheDefaultExpire</td>
<td>3600 (1)</td>
</tr>
<tr>
<td>CacheDirLength</td>
<td>2</td>
</tr>
<tr>
<td>CacheDirLevels</td>
<td>3</td>
</tr>
<tr>
<td>CacheDisable url-string</td>
<td>URL</td>
</tr>
<tr>
<td>CacheEnable cache_type url-string</td>
<td></td>
</tr>
<tr>
<td>CacheFile file-path [file-path] ...</td>
<td></td>
</tr>
<tr>
<td>Cache a list of file handles at startup time</td>
<td></td>
</tr>
<tr>
<td>CachefileCacheControl</td>
<td>Off</td>
</tr>
<tr>
<td>CachefileIgnoreCacheControlOnOff</td>
<td>Off</td>
</tr>
<tr>
<td>CachefileIgnoreHeaders header-string [header-string] ...</td>
<td>None</td>
</tr>
<tr>
<td><strong>string</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>HTTP</td>
<td></td>
</tr>
<tr>
<td>CacheIgnoreNoLastMod On</td>
<td>Off</td>
</tr>
<tr>
<td>Last Modified</td>
<td></td>
</tr>
<tr>
<td>CacheIgnoreQueryString On</td>
<td>Off</td>
</tr>
<tr>
<td>Ignore query string when caching</td>
<td></td>
</tr>
<tr>
<td>CacheIgnoreURLSessionIdentifiers identifier</td>
<td>None</td>
</tr>
<tr>
<td>[identifier]</td>
<td>Ignore defined session identifiers encoded in the URL when caching</td>
</tr>
<tr>
<td>CacheLastModifiedFactor float</td>
<td>0.1</td>
</tr>
<tr>
<td>LastModified (expiry)</td>
<td></td>
</tr>
<tr>
<td>CacheLock on</td>
<td>off</td>
</tr>
<tr>
<td>Enable the thundering herd lock.</td>
<td></td>
</tr>
<tr>
<td>CacheLockMaxAge integer</td>
<td>5</td>
</tr>
<tr>
<td>Set the maximum possible age of a cache lock.</td>
<td></td>
</tr>
<tr>
<td>CacheLockPath directory</td>
<td>/tmp/mod_cache-lock</td>
</tr>
<tr>
<td>Set the lock path directory.</td>
<td></td>
</tr>
<tr>
<td>CacheMaxExpire seconds</td>
<td>86400 ()</td>
</tr>
<tr>
<td>CacheMaxFileSize bytes</td>
<td>1000000</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CacheMinFileSize bytes</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CacheNegotiatedDocs On</td>
<td>Off</td>
</tr>
<tr>
<td>CacheRoot directory</td>
<td></td>
</tr>
<tr>
<td>CacheStoreNoStore On</td>
<td>Off</td>
</tr>
<tr>
<td>no-store</td>
<td></td>
</tr>
<tr>
<td>CacheStorePrivate On</td>
<td>Off</td>
</tr>
<tr>
<td>private</td>
<td></td>
</tr>
<tr>
<td>CGIDScriptTimeout time[s</td>
<td>ms]</td>
</tr>
<tr>
<td>The length of time to wait for more output from the CGI program</td>
<td></td>
</tr>
<tr>
<td>CGIMapExtension cgi-path .extension</td>
<td>CGI</td>
</tr>
<tr>
<td>Configuration</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>CharsetDefault charset</strong></td>
<td>Charset to translate into</td>
</tr>
<tr>
<td><strong>CharsetOptions option [option] ...</strong></td>
<td>Configures charset translation behavior</td>
</tr>
<tr>
<td><strong>CharsetSourceEnc charset</strong></td>
<td>Source charset of files</td>
</tr>
<tr>
<td>**CheckCaseOnly on</td>
<td>off**</td>
</tr>
<tr>
<td>**CheckSpelling on</td>
<td>off**</td>
</tr>
<tr>
<td><strong>ChrootDir /path/to/directory</strong></td>
<td>Directory for apache to run chroot(8) after startup.</td>
</tr>
<tr>
<td>**ContentDigest On</td>
<td>Off**</td>
</tr>
<tr>
<td><strong>CookieDomain domain</strong></td>
<td>The domain to which the tracking cookie applies</td>
</tr>
<tr>
<td><strong>CookieExpires expiry-period</strong></td>
<td>Expiry time for the tracking cookie</td>
</tr>
<tr>
<td><strong>CookieLog filename</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CookieName token</strong></td>
<td>Name of the tracking cookie</td>
</tr>
<tr>
<td><strong>CookieStyle</strong></td>
<td>Netscape</td>
</tr>
<tr>
<td>**Netscape</td>
<td>Cookie</td>
</tr>
<tr>
<td>**CookieTracking on</td>
<td>off**</td>
</tr>
<tr>
<td><strong>CoreDumpDirectory directory</strong></td>
<td>Apache</td>
</tr>
<tr>
<td>**CustomLog file</td>
<td>pipe format</td>
</tr>
<tr>
<td>**Dav On</td>
<td>Off</td>
</tr>
<tr>
<td>**DavDepthInfinity on</td>
<td>off**</td>
</tr>
</tbody>
</table>

| **CoreDumpDirectory directory** | Apache |
| **CustomLog file|pipe format|nickname [env= [!]environment-variable]** | |

| **Dav On|Off|provider-name** | WebDAV HTTP | Off |
| **DavDepthInfinity on|off** | |

<p>| <strong>CharsetSourceEnc charset</strong> | Source charset of files |
| PROPFIND, Depth: Infinity | DavGenericLockDB file-path | DavLockDB file-path | DavMinTimeout seconds 0 | DBDExptime time-in-seconds 300 | DBDKeep number 2 | DBDMax number 10 | DBDMin number 1 | DBDParams param1=value1[,param2=value2] Parameters for database connection | DBDPersist On|Off Whether to use persistent connections | DBDPrepareSQL &quot;SQL statement&quot; label Define an SQL prepared statement | DBDriver name Specify an SQL driver | DefaultIcon url-path | DefaultLanguage MIME-lang | DefaultType MIME-type|none MIME text/plain | DeflateBufferSize value 8096 zlib | DeflateCompressionLevel value | DeflateFilterNote [type] notename |</p>
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflateInflateLimitRequestBody</td>
<td>Maximum size of inflated request bodies</td>
<td></td>
</tr>
<tr>
<td>DeflateInflateRatioBurst value</td>
<td>Maximum number of times the inflation ratio for request bodies can be crossed</td>
<td></td>
</tr>
<tr>
<td>DeflateInflateRatioLimit value</td>
<td>Maximum inflation ratio for request bodies</td>
<td></td>
</tr>
<tr>
<td>DeflateMemLevel value</td>
<td>9</td>
<td>zlib</td>
</tr>
<tr>
<td>DeflateWindowSize value</td>
<td>15</td>
<td>Zlib</td>
</tr>
</tbody>
</table>
| Deny from all|host|env=env-variable |[
<p>| [host|env=env-variable]] ... |
| &lt;Directory directory-path&gt; ... &lt;/Directory&gt; |
| DirectoryIndex local-url [local-url] ... | index.html |
| &lt;DirectoryMatch regex&gt; ... &lt;/DirectoryMatch&gt; |
| DirectorySlash On|Off | On |
| DocumentRoot directory-path | /usr/local/apache/h + |
| DumpIOInput On|Off | Off |
| DumpIOLogLevel level | debug |
| Controls the logging level of the DumpIO output |
| DumpIOOutput On|Off | Off |
| EnableExceptionHook On|Off | Off |
| EnableMMAP On|Off | On |
| EnableSendfile On|Off | On |
| sendfile |</p>
<table>
<thead>
<tr>
<th>ErrorDocument</th>
<th>error-code document</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorLog</td>
<td>file-path</td>
</tr>
<tr>
<td>Example</td>
<td>Demonstration directive to illustrate the Apache module API</td>
</tr>
<tr>
<td>ExpiresActive</td>
<td>On</td>
</tr>
<tr>
<td>Expires</td>
<td></td>
</tr>
<tr>
<td>ExpiresByType</td>
<td>MIME-type &lt;code&gt;seconds</td>
</tr>
<tr>
<td>MIME Expires</td>
<td></td>
</tr>
<tr>
<td>ExpiresDefault</td>
<td>&lt;code&gt;seconds</td>
</tr>
<tr>
<td>ExtendedStatus</td>
<td>On</td>
</tr>
<tr>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>ExtFilterDefine</td>
<td>filtername parameters</td>
</tr>
<tr>
<td>ExtFilterOptions</td>
<td>option [option] ...</td>
</tr>
<tr>
<td>DebugLevel=0 NoLog +</td>
<td></td>
</tr>
<tr>
<td>mod_ext_filter</td>
<td></td>
</tr>
<tr>
<td>FallbackResource</td>
<td>disabled</td>
</tr>
<tr>
<td>Define a default URL for requests that don't map to a file</td>
<td></td>
</tr>
<tr>
<td>FileETag</td>
<td>component ...</td>
</tr>
<tr>
<td>ETag HTTP</td>
<td></td>
</tr>
<tr>
<td>&lt;Files filename&gt; ... &lt;/Files&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;FilesMatch regex&gt; ... &lt;/FilesMatch&gt;</td>
<td></td>
</tr>
<tr>
<td>FilterChain</td>
<td>[+</td>
</tr>
<tr>
<td>Configure the filter chain</td>
<td></td>
</tr>
<tr>
<td>FilterDeclare filter-name [type]</td>
<td></td>
</tr>
<tr>
<td>Declare a smart filter</td>
<td></td>
</tr>
<tr>
<td>FilterProtocol</td>
<td>filter-name [provider-name] proto-flags</td>
</tr>
<tr>
<td>Deal with correct HTTP protocol handling</td>
<td></td>
</tr>
<tr>
<td>FilterProvider</td>
<td>filter-name provider-name</td>
</tr>
<tr>
<td>Configuration</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>`[req</td>
<td>resp</td>
</tr>
<tr>
<td><code>FilterTrace filter-name level</code></td>
<td>Get debug/diagnostic information from <code>mod_filter</code></td>
</tr>
<tr>
<td>`ForceLanguagePriority None</td>
<td>Prefer</td>
</tr>
<tr>
<td>`ForceType MIME-type</td>
<td>None`</td>
</tr>
<tr>
<td>`ForensicLog filename</td>
<td>pipe`</td>
</tr>
<tr>
<td><code>GprofDir /tmp/gprof/%</code></td>
<td>Directory to write gmon.out profiling data to.</td>
</tr>
<tr>
<td><code>GracefulShutDownTimeout seconds</code></td>
<td></td>
</tr>
<tr>
<td><code>Group unix-group</code></td>
<td><code>-1</code></td>
</tr>
<tr>
<td>`Header [condition] set</td>
<td>append</td>
</tr>
<tr>
<td>`HostnameLookups On</td>
<td>Off</td>
</tr>
<tr>
<td>`HttpProtocolOptions [Strict</td>
<td>Unsafe] [RegisteredMethods</td>
</tr>
<tr>
<td>`IdentityCheck On</td>
<td>Off`</td>
</tr>
<tr>
<td><code>IdentityCheckTimeout seconds</code></td>
<td>30</td>
</tr>
<tr>
<td><code>Ident &lt;IfDefine [!]parameter-name&gt; ... &lt;/IfDefine&gt;</code></td>
<td></td>
</tr>
<tr>
<td>`&lt;IfModule [!]module-file</td>
<td>module-identifier&gt; ...`</td>
</tr>
<tr>
<td><strong>&lt;IfModule&gt;</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>&lt;IfVersion ![operator] version&gt; ...</strong></td>
<td></td>
</tr>
<tr>
<td><strong>&lt;/IfVersion&gt;</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **ImapBase map|referer|URL** | **http://servername/**
Default base for imagemap files |
| **ImapDefault error|nocontent|map|referer|URL** | **nocontent**
Default action when an imagemap is called with coordinates that are not explicitly mapped |
| **ImapMenu** | **none|formatted|semiformatted|unformatted**
Action if no coordinates are given when calling an imagemap |
| **Include file-path|directory-path** |  |
| **IndexHeadInsert "markup ..."** |  
Inserts text in the HEAD section of an index page. |
| **IndexIgnore file [file] ...** |  |
| **IndexOptions [+|-]option [[+|-]option] ...** |  |
| **IndexOrderDefault Ascending|Descending Name|Date|Size|Description** | **Ascending Name** |
| **IndexStyleSheet url-path** | **CSS** |
| **ISAPIAppendLogToErrors on|off** | **off**
Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log |
| **ISAPIAppendLogToQuery on|off** | **on**
Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field |
| **ISAPICacheFile file-path [file-path] ...** |  
ISAPI .dll files to be loaded at startup |
| **ISAPIFakeAsync on|off** | **off**
Fake asynchronous support for ISAPI callbacks |
| **ISAPILogNotSupported on|off** | **off**
Log unsupported feature requests from ISAPI extensions |
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ISAPIReadAheadBuffer size</strong></td>
<td>49152</td>
</tr>
<tr>
<td>Size of the Read Ahead Buffer sent to ISAPI extensions</td>
<td></td>
</tr>
<tr>
<td><strong>KeepAlive</strong></td>
<td>On</td>
</tr>
<tr>
<td><strong>HTTP</strong></td>
<td></td>
</tr>
<tr>
<td><strong>KeepAliveTimeout seconds</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>LanguagePriority MIME-lang [MIME-lang] ... variant</strong></td>
<td></td>
</tr>
<tr>
<td><strong>LDAPCacheEntries number</strong></td>
<td>1024</td>
</tr>
<tr>
<td>Maximum number of entries in the primary LDAP cache</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPCacheTTL seconds</strong></td>
<td>600</td>
</tr>
<tr>
<td>Time that cached items remain valid</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPConnectionTimeout seconds</strong></td>
<td></td>
</tr>
<tr>
<td>Specifies the socket connection timeout in seconds</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPOpCacheEntries number</strong></td>
<td>1024</td>
</tr>
<tr>
<td>Number of entries used to cache LDAP compare operations</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPOpCacheTTL seconds</strong></td>
<td>600</td>
</tr>
<tr>
<td>Time that entries in the operation cache remain valid</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPSharedCacheFile directory-path/filename</strong></td>
<td></td>
</tr>
<tr>
<td>Sets the shared memory cache file</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPSharedCacheSize bytes</strong></td>
<td>500000</td>
</tr>
<tr>
<td>Size in bytes of the shared-memory cache</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPTrustedClientCert type directory-path/filename/nickname [password]</strong></td>
<td></td>
</tr>
<tr>
<td>Sets the file containing or nickname referring to a per connection client certificate. Not all LDAP toolkits support per connection client certificates.</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPTrustedGlobalCert type directory-path/filename [password]</strong></td>
<td></td>
</tr>
<tr>
<td>Sets the file or database containing global trusted Certificate Authority or global client certificates.</td>
<td></td>
</tr>
<tr>
<td><strong>LDAPTrustedMode type</strong></td>
<td></td>
</tr>
<tr>
<td>Specifies the SSL/TLS mode to be used when connecting to an LDAP server.</td>
<td></td>
</tr>
<tr>
<td>**LDAPVerifyServerCert On</td>
<td>Off**</td>
</tr>
<tr>
<td>Force server certificate verification</td>
<td></td>
</tr>
<tr>
<td><strong>&lt;Limit method [method] ... &gt; ... &lt;/Limit&gt;</strong></td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>&lt;LimitExcept method [method] ... &gt; ...</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>HTTP</td>
<td>&lt;/LimitExcept&gt;</td>
</tr>
<tr>
<td>HTTP</td>
<td>LimitInternalRecursion number [number] 10</td>
</tr>
<tr>
<td>HTTP</td>
<td>LimitRequestBody bytes 0</td>
</tr>
<tr>
<td>HTTP</td>
<td>LimitRequestFields number 100</td>
</tr>
<tr>
<td>HTTP</td>
<td>LimitRequestFieldSize bytes 8190</td>
</tr>
<tr>
<td>HTTP</td>
<td>LimitRequestLine bytes 8190</td>
</tr>
<tr>
<td>HTTP</td>
<td>LimitXMLRequestBody bytes 1000000</td>
</tr>
<tr>
<td>XML</td>
<td>Listen [IP-address:]portnumber [protocol]</td>
</tr>
<tr>
<td>listen IP</td>
<td>ListenBacklog backlog</td>
</tr>
<tr>
<td>LoadFile filename [filename] ...</td>
<td></td>
</tr>
<tr>
<td>LoadModule module filename</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>&lt;Location URL-path</td>
</tr>
<tr>
<td>URL</td>
<td>&lt;LocationMatch regex&gt; ... &lt;/LocationMatch&gt;</td>
</tr>
<tr>
<td>URL</td>
<td>LockFile filename logs/accept.lock</td>
</tr>
<tr>
<td>LogFormat format</td>
<td>nickname [nickname] &quot;%h %l %u %t %r\n&quot; +</td>
</tr>
<tr>
<td>LogLevel level warn</td>
<td></td>
</tr>
<tr>
<td>MaxClients number</td>
<td></td>
</tr>
<tr>
<td>ErrorLog</td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MaxKeepAliveRequests</strong></td>
<td>number of requests allowed before returning the complete resource</td>
<td>100</td>
</tr>
<tr>
<td><strong>MaxMemFree</strong></td>
<td>KBytes free()</td>
<td>0</td>
</tr>
<tr>
<td><strong>MaxRanges</strong></td>
<td>default</td>
<td>unlimited</td>
</tr>
<tr>
<td><strong>MaxRequestsPerChild</strong></td>
<td>number</td>
<td>10000</td>
</tr>
<tr>
<td><strong>MaxRequestsPerThread</strong></td>
<td>number</td>
<td>0</td>
</tr>
<tr>
<td><strong>MaxSpareServers</strong></td>
<td>number</td>
<td>10</td>
</tr>
<tr>
<td><strong>MaxSpareThreads</strong></td>
<td>number</td>
<td>100</td>
</tr>
<tr>
<td><strong>MaxThreads</strong></td>
<td>number</td>
<td>2048</td>
</tr>
<tr>
<td><strong>MCacheMaxObjectCount</strong></td>
<td>value</td>
<td>1009</td>
</tr>
<tr>
<td><strong>MCacheMaxObjectSize</strong></td>
<td>bytes</td>
<td>10000</td>
</tr>
<tr>
<td><strong>MCacheMaxStreamingBuffer</strong></td>
<td>size_in_bytes</td>
<td>of 100000 MCacheM +</td>
</tr>
<tr>
<td><strong>MCacheMinObjectSize</strong></td>
<td>bytes</td>
<td>0</td>
</tr>
<tr>
<td><strong>MCacheRemovalAlgorithm</strong></td>
<td>LRU</td>
<td>GDSF</td>
</tr>
<tr>
<td><strong>MCacheSize</strong></td>
<td>KBytes</td>
<td>100</td>
</tr>
<tr>
<td><strong>MergeTrailers</strong></td>
<td>[on</td>
<td>off]</td>
</tr>
<tr>
<td><strong>MetaDir</strong></td>
<td>directory</td>
<td>.web</td>
</tr>
<tr>
<td><strong>Name of the directory to find CERN-style meta information files</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Notes:*
- MCacheMaxObjectSize is set to 0.
- MergeTrailers is set to off.
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaFiles</td>
<td>on/off</td>
</tr>
<tr>
<td>MetaSuffix</td>
<td>suffix</td>
</tr>
<tr>
<td>MimeMagicFile</td>
<td>file-path</td>
</tr>
<tr>
<td>MinSpareServers</td>
<td>number</td>
</tr>
<tr>
<td>MMapFile</td>
<td>file-path</td>
</tr>
<tr>
<td>ModMimeUsePathInfo</td>
<td>On</td>
</tr>
<tr>
<td>MultiviewsMatch</td>
<td>Any</td>
</tr>
<tr>
<td>NameVirtualHost</td>
<td>addr[:port]</td>
</tr>
<tr>
<td>NoProxy</td>
<td>host [host]</td>
</tr>
<tr>
<td>NWSSLTrustedCerts</td>
<td>filename [filename]</td>
</tr>
<tr>
<td>NWSSLUpgradeable</td>
<td>[IP-address:]portnumber</td>
</tr>
<tr>
<td>Options</td>
<td>[+</td>
</tr>
<tr>
<td>Order</td>
<td>ordering</td>
</tr>
<tr>
<td>PassEnv</td>
<td>env-variable [env-variable]</td>
</tr>
<tr>
<td>PidFile</td>
<td>filename</td>
</tr>
<tr>
<td>Protocol</td>
<td>protocol</td>
</tr>
<tr>
<td>Protocol for a listening socket</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>ProtocolEcho</strong></td>
<td>On</td>
</tr>
<tr>
<td><strong>&lt;Proxy wildcard-url&gt; ...&lt;/Proxy&gt;</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ProxyBadHeader</strong></td>
<td>IsError</td>
</tr>
<tr>
<td><strong>ProxyBlock</strong></td>
<td>*word</td>
</tr>
<tr>
<td><strong>ProxyDomain</strong></td>
<td>Domain</td>
</tr>
<tr>
<td><strong>ProxyErrorOverride</strong></td>
<td>On</td>
</tr>
<tr>
<td><strong>ProxyFtpDirCharset</strong></td>
<td>character set</td>
</tr>
<tr>
<td><strong>ProxyIOBufferSize</strong></td>
<td>bytes</td>
</tr>
<tr>
<td><strong>&lt;ProxyMatch regex&gt; ...&lt;/ProxyMatch&gt;</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ProxyMaxForwards</strong></td>
<td>number</td>
</tr>
<tr>
<td><strong>ProxyPass</strong></td>
<td>[path] !</td>
</tr>
<tr>
<td><strong>ProxyPassInterpolateEnv</strong></td>
<td>On</td>
</tr>
<tr>
<td><strong>ProxyPassMatch</strong></td>
<td>[regex] !</td>
</tr>
<tr>
<td><strong>ProxyPassReverse</strong></td>
<td>[path] url [interpolate]</td>
</tr>
<tr>
<td><strong>ProxyPassReverseCookieDomain</strong></td>
<td>internal-domain public-domain [interpolate]</td>
</tr>
<tr>
<td>Configuration</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>ProxyPassReverseCookiePath <code>internal-path</code> <code>public-path [interpolate]</code></td>
<td></td>
</tr>
<tr>
<td>Set-Cookie Path</td>
<td></td>
</tr>
<tr>
<td>ProxyPreserveHost `On</td>
<td>Off`</td>
</tr>
<tr>
<td>Host HTTP</td>
<td></td>
</tr>
<tr>
<td>ProxyReceiveBufferSize <code>bytes</code></td>
<td>0</td>
</tr>
<tr>
<td>HTTP FTP</td>
<td></td>
</tr>
<tr>
<td>ProxyRemote <code>match remote-server</code></td>
<td></td>
</tr>
<tr>
<td>ProxyRemoteMatch <code>regex remote-server</code></td>
<td></td>
</tr>
<tr>
<td>ProxyRequests `On</td>
<td>Off`</td>
</tr>
<tr>
<td><code>()</code></td>
<td></td>
</tr>
<tr>
<td>ProxySCGIInternalRedirect `On</td>
<td>Off`</td>
</tr>
<tr>
<td>ProxySCGISendfile `On</td>
<td>Off</td>
</tr>
<tr>
<td><code>X-Sendfile</code></td>
<td></td>
</tr>
<tr>
<td>ProxySet <code>url key=value [key=value ...]</code></td>
<td></td>
</tr>
<tr>
<td>ProxyStatus `Off</td>
<td>On</td>
</tr>
<tr>
<td><code>mod_status</code></td>
<td></td>
</tr>
<tr>
<td>ProxyTimeout <code>seconds</code></td>
<td></td>
</tr>
<tr>
<td>ProxyVia `On</td>
<td>Off</td>
</tr>
<tr>
<td><code>Via HTTP</code></td>
<td></td>
</tr>
<tr>
<td>ReadmeName <code>filename</code></td>
<td></td>
</tr>
<tr>
<td>ReceiveBufferSize <code>bytes</code></td>
<td>0</td>
</tr>
<tr>
<td>TCP</td>
<td></td>
</tr>
<tr>
<td>Redirect <code>[status] URL-path URL</code></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
</tr>
<tr>
<td>RedirectMatch <code>[status] regex URL</code></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
</tr>
<tr>
<td>RedirectPermanent <code>URL-path URL</code></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>RedirectTemp URL-path URL</code></td>
<td>Redirect Temporary URL to URL</td>
</tr>
<tr>
<td><code>RegisterHttpMethod method [method [...]]</code></td>
<td>Register non-standard HTTP methods</td>
</tr>
<tr>
<td><code>RemoveCharset extension [extension]</code></td>
<td>Remove charset extension</td>
</tr>
<tr>
<td><code>RemoveEncoding extension [extension]</code></td>
<td>Remove encoding extension</td>
</tr>
<tr>
<td><code>RemoveHandler extension [extension]</code></td>
<td>Remove handler extension</td>
</tr>
<tr>
<td><code>RemoveInputFilter extension [extension]</code></td>
<td>Remove input filter extension</td>
</tr>
<tr>
<td><code>RemoveLanguage extension [extension]</code></td>
<td>Remove language extension</td>
</tr>
<tr>
<td><code>RemoveOutputFilter extension [extension]</code></td>
<td>Remove output filter extension</td>
</tr>
<tr>
<td><code>RemoveType extension [extension]</code></td>
<td>Remove type extension</td>
</tr>
<tr>
<td>`RequestHeader set</td>
<td>append</td>
</tr>
<tr>
<td><code>RequestReadTimeout [header=timeout[-maxtimeout],MinRate=rate] [body=timeout[-maxtimeout],MinRate=rate]</code></td>
<td>Set timeout values for receiving request headers and body from client</td>
</tr>
<tr>
<td><code>Require entity-name [entity-name]</code></td>
<td>Require entity name</td>
</tr>
<tr>
<td><code>RewriteBase URL-path</code></td>
<td>Sets the base URL for per-directory rewrites</td>
</tr>
<tr>
<td><code>RewriteCond TestString CondPattern</code></td>
<td>Define a condition under which rewriting will take place</td>
</tr>
<tr>
<td>`RewriteEngine on</td>
<td>off`</td>
</tr>
<tr>
<td><code>RewriteLock file-path</code></td>
<td>Set lock file name for <code>RewriteMap</code> synchronization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>off</td>
</tr>
</tbody>
</table>

Note: `RewriteEngine` is set to `off` in the example.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RewriteLog `file-path</td>
<td>pipe`</td>
</tr>
<tr>
<td>RewriteLogLevel <code>Level</code></td>
<td>Sets the verbosity of the log file used by the rewrite engine</td>
</tr>
<tr>
<td>RewriteMap <code>MapName MapType:MapSource</code></td>
<td>Defines a mapping function for key-lookup</td>
</tr>
<tr>
<td>RewriteOptions <code>Options</code></td>
<td>Sets some special options for the rewrite engine</td>
</tr>
<tr>
<td>RewriteRule <code>Pattern Substitution [flags]</code></td>
<td>Defines rules for the rewriting engine</td>
</tr>
<tr>
<td>RLimitCPU `seconds</td>
<td>max [seconds</td>
</tr>
<tr>
<td>RLimitMEM `bytes</td>
<td>max [bytes</td>
</tr>
<tr>
<td>RLimitNPROC `number</td>
<td>max [number</td>
</tr>
<tr>
<td>Satisfy `Any</td>
<td>All`</td>
</tr>
<tr>
<td>ScoreBoardFile <code>file-path</code></td>
<td>logs/apache_status</td>
</tr>
<tr>
<td>Script <code>method cgi-script</code></td>
<td>CGI</td>
</tr>
<tr>
<td>ScriptAlias `URL-path file-path</td>
<td>directory-path`</td>
</tr>
<tr>
<td>ScriptAliasMatch `regex file-path</td>
<td>directory-path`</td>
</tr>
<tr>
<td>ScriptInterpreterSource `Registry</td>
<td>Registry-Strict</td>
</tr>
<tr>
<td>ScriptLog <code>file-path</code></td>
<td>CGI</td>
</tr>
<tr>
<td>ScriptLogBuffer <code>bytes</code></td>
<td>1024</td>
</tr>
<tr>
<td>ScriptLogLength <code>bytes</code></td>
<td>10385760</td>
</tr>
<tr>
<td>Configuration Item</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>ScriptSock file-path</td>
<td>logs/cgisock</td>
</tr>
<tr>
<td>SecureListen [IP-address:]portnumber</td>
<td></td>
</tr>
<tr>
<td>Certificate-Name [MUTUAL]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enables SSL encryption for the specified port</td>
</tr>
<tr>
<td>SeeRequestTail On</td>
<td>Off</td>
</tr>
<tr>
<td>SendBufferSize bytes</td>
<td>0</td>
</tr>
<tr>
<td>ServerAdmin email-address</td>
<td>URL</td>
</tr>
<tr>
<td>ServerAlias hostname [hostname]...</td>
<td></td>
</tr>
<tr>
<td>ServerLimit number</td>
<td></td>
</tr>
<tr>
<td>ServerName [scheme://]fully-qualified-domain-name[:port]</td>
<td></td>
</tr>
<tr>
<td>ServerPath URL-path</td>
<td></td>
</tr>
<tr>
<td>ServerRoot directory-path</td>
<td>/usr/local/apache</td>
</tr>
<tr>
<td>ServerSignature On</td>
<td>Off</td>
</tr>
<tr>
<td>ServerTokens</td>
<td>Full</td>
</tr>
<tr>
<td></td>
<td>Major</td>
</tr>
<tr>
<td></td>
<td>Server HTTP</td>
</tr>
<tr>
<td>SetEnv env-variable value</td>
<td></td>
</tr>
<tr>
<td>SetEnvIf attribute regex [!]env-variable=value [!]env-variable=value] ...</td>
<td></td>
</tr>
<tr>
<td>SetEnvIfNoCase attribute regex [!]env-variable=value [!]env-variable=value] ...</td>
<td></td>
</tr>
<tr>
<td>Configuration Item</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>SetHandler</code> `handler-name</td>
<td>None`</td>
</tr>
<tr>
<td><code>SetInputFilter</code> <code>filter[:filter...]</code></td>
<td>Enables content filtering for input data</td>
</tr>
<tr>
<td><code>SetOutputFilter</code> <code>filter[:filter...]</code></td>
<td>Enables content filtering for output data</td>
</tr>
<tr>
<td><code>SSIEnableAccess</code> `on</td>
<td>off`</td>
</tr>
<tr>
<td><code>SSIEndTag</code> <code>tag</code></td>
<td>Include processing for SSI end tags</td>
</tr>
<tr>
<td><code>SSIErrorMsg</code> <code>message</code></td>
<td>SSI error message processing</td>
</tr>
<tr>
<td><code>SSIETag</code> `on</td>
<td>off`</td>
</tr>
<tr>
<td><code>SSILastModified</code> `on</td>
<td>off`</td>
</tr>
<tr>
<td><code>SSIStartTag</code> <code>tag</code></td>
<td>Include processing for SSI start tags</td>
</tr>
<tr>
<td><code>SSITimeFormat</code> <code>formatstring</code></td>
<td>Time format string for SSI processing</td>
</tr>
<tr>
<td><code>SSIUndefinedEcho</code> <code>string</code></td>
<td>Echo processing for SSI undefined strings</td>
</tr>
<tr>
<td><code>SSLCACertificateFile</code> <code>file-path</code></td>
<td>File of concatenated PEM-encoded CA Certificates for Client Auth</td>
</tr>
<tr>
<td><code>SSLCACertificatePath</code> <code>directory-path</code></td>
<td>Directory of PEM-encoded CA Certificates for Client Auth</td>
</tr>
<tr>
<td><code>SSLCADNRequestFile</code> <code>file-path</code></td>
<td>File of concatenated PEM-encoded CA Certificates for defining acceptable CA names</td>
</tr>
<tr>
<td><code>SSLCADNRequestPath</code> <code>directory-path</code></td>
<td>Directory of PEM-encoded CA Certificates for defining acceptable CA names</td>
</tr>
<tr>
<td><code>SSLCARevocationFile</code> <code>file-path</code></td>
<td>File of concatenated PEM-encoded CA CRLs for Client Auth</td>
</tr>
<tr>
<td><code>SSLCARevocationPath</code> <code>directory-path</code></td>
<td>Directory of PEM-encoded CA CRLs for Client Auth</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SSLCertificateChainFile file-path</td>
<td>File of PEM-encoded Server CA Certificates</td>
</tr>
<tr>
<td>SSLCertificateFile file-path</td>
<td>Server PEM-encoded X.509 Certificate file</td>
</tr>
<tr>
<td>SSLCertificateKeyFile file-path</td>
<td>Server PEM-encoded Private Key file</td>
</tr>
<tr>
<td>SSLCipherSuite cipher-spec</td>
<td>Cipher Suite available for negotiation in SSL handshake</td>
</tr>
<tr>
<td>SSLCompression on</td>
<td>off</td>
</tr>
<tr>
<td>SSLCryptoDevice engine</td>
<td>Enable use of a cryptographic hardware accelerator</td>
</tr>
<tr>
<td>SSLEngine on</td>
<td>off</td>
</tr>
<tr>
<td>SSLFIPS on</td>
<td>off</td>
</tr>
<tr>
<td>SSLHonorCipherOrder flag</td>
<td>Option to prefer the server's cipher preference order</td>
</tr>
<tr>
<td>SSLInsecureRenegotiation flag</td>
<td>Option to enable support for insecure renegotiation</td>
</tr>
<tr>
<td>SSLMutex type</td>
<td>Semaphore for internal mutual exclusion of operations</td>
</tr>
<tr>
<td>SSLOptions [+</td>
<td>-]option ...</td>
</tr>
<tr>
<td>SSLPassPhraseDialog type</td>
<td>Type of pass phrase dialog for encrypted private keys</td>
</tr>
<tr>
<td>SSLProtocol [+</td>
<td>-]protocol ...</td>
</tr>
<tr>
<td>SSLProxyCACertificateFile file-path</td>
<td>File of concatenated PEM-encoded CA Certificates for Remote Server Auth</td>
</tr>
<tr>
<td>SSLProxyCACertificatePath directory-path</td>
<td>Directory of PEM-encoded CA Certificates for Remote Server Auth</td>
</tr>
<tr>
<td>SSLProxyCARevocationFile file-path</td>
<td>File of concatenated PEM-encoded CA CRLs for Remote Server Auth</td>
</tr>
<tr>
<td>Configuration</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>SSLProxyCARevocationPath</td>
<td>directory-path</td>
</tr>
<tr>
<td>Directory of PEM-encoded CA CRLs for Remote Server Auth</td>
<td></td>
</tr>
<tr>
<td>SSLProxyCheckPeerCN</td>
<td>on</td>
</tr>
<tr>
<td>Whether to check the remote server certificates CN field</td>
<td></td>
</tr>
<tr>
<td>SSLProxyCheckPeerExpire</td>
<td>on</td>
</tr>
<tr>
<td>Whether to check if remote server certificate is expired</td>
<td></td>
</tr>
<tr>
<td>SSLProxyCipherSuite cipher-spec</td>
<td>ALL:!ADH:RC4+RSA:+</td>
</tr>
<tr>
<td>Cipher Suite available for negotiation in SSL proxy handshake</td>
<td></td>
</tr>
<tr>
<td>SSLProxyEngine</td>
<td>on</td>
</tr>
<tr>
<td>SSL Proxy Engine Operation Switch</td>
<td></td>
</tr>
<tr>
<td>SSLProxyMachineCertificateChainFile filename</td>
<td></td>
</tr>
<tr>
<td>File of concatenated PEM-encoded CA certificates to be used by the proxy for choosing a certificate</td>
<td></td>
</tr>
<tr>
<td>SSLProxyMachineCertificateFile filename</td>
<td></td>
</tr>
<tr>
<td>File of concatenated PEM-encoded client certificates and keys to be used by the proxy</td>
<td></td>
</tr>
<tr>
<td>SSLProxyMachineCertificatePath directory</td>
<td></td>
</tr>
<tr>
<td>Directory of PEM-encoded client certificates and keys to be used by the proxy</td>
<td></td>
</tr>
<tr>
<td>SSLProxyProtocol [+]protocol</td>
<td>all</td>
</tr>
<tr>
<td>Configure usable SSL protocol flavors for proxy usage</td>
<td></td>
</tr>
<tr>
<td>SSLProxyVerify level</td>
<td>none</td>
</tr>
<tr>
<td>Type of remote server Certificate verification</td>
<td></td>
</tr>
<tr>
<td>SSLProxyVerifyDepth number</td>
<td>1</td>
</tr>
<tr>
<td>Maximum depth of CA Certificates in Remote Server Certificate verification</td>
<td></td>
</tr>
<tr>
<td>SSLRandomSeed context source [bytes]</td>
<td></td>
</tr>
<tr>
<td>Pseudo Random Number Generator (PRNG) seeding source</td>
<td></td>
</tr>
<tr>
<td>SSLRenegBufferSize bytes</td>
<td>131072</td>
</tr>
<tr>
<td>Set the size for the SSL renegotiation buffer</td>
<td></td>
</tr>
<tr>
<td>SSLRequire expression</td>
<td></td>
</tr>
<tr>
<td>Allow access only when an arbitrarily complex boolean expression is true</td>
<td></td>
</tr>
<tr>
<td>SSLRequireSSL</td>
<td></td>
</tr>
<tr>
<td>Deny access when SSL is not used for the HTTP request</td>
<td></td>
</tr>
<tr>
<td>SSLSessionCache type</td>
<td>none</td>
</tr>
<tr>
<td>Type of the global/inter-process SSL Session Cache</td>
<td></td>
</tr>
<tr>
<td>SSLSessionCacheTimeout seconds</td>
<td>300</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Number of seconds before an SSL session expires in the Session Cache</td>
<td></td>
</tr>
<tr>
<td>SSLSessionTicketKeyFile file-path</td>
<td>Persistent encryption/decryption key for TLS session tickets</td>
</tr>
<tr>
<td>SSLSessionTickets on</td>
<td>off</td>
</tr>
<tr>
<td>Enable or disable use of TLS session tickets</td>
<td></td>
</tr>
<tr>
<td>SSLStrictSNIVHostCheck on</td>
<td>off</td>
</tr>
<tr>
<td>Whether to allow non SNI clients to access a name based virtual host.</td>
<td></td>
</tr>
<tr>
<td>SSLUserName varname</td>
<td></td>
</tr>
<tr>
<td>Variable name to determine user name</td>
<td></td>
</tr>
<tr>
<td>SSLVerifyClient level</td>
<td>none</td>
</tr>
<tr>
<td>Type of Client Certificate verification</td>
<td></td>
</tr>
<tr>
<td>SSLVerifyDepth number</td>
<td>1</td>
</tr>
<tr>
<td>Maximum depth of CA Certificates in Client Certificate verification</td>
<td></td>
</tr>
<tr>
<td>StartServers number</td>
<td></td>
</tr>
<tr>
<td>StartThreads number</td>
<td></td>
</tr>
<tr>
<td>Substitute s/pattern/substitution/[infq]</td>
<td>Pattern to filter the response content</td>
</tr>
<tr>
<td>SubstituteInheritBefore on</td>
<td>off</td>
</tr>
<tr>
<td>Change the merge order of inherited patterns</td>
<td></td>
</tr>
<tr>
<td>Suexec On</td>
<td>Off</td>
</tr>
<tr>
<td>Enable or disable the suEXEC feature</td>
<td></td>
</tr>
<tr>
<td>SuexecUserGroup User Group CGI</td>
<td></td>
</tr>
<tr>
<td>ThreadLimit number</td>
<td></td>
</tr>
<tr>
<td>ThreadsPerChild number</td>
<td></td>
</tr>
<tr>
<td>ThreadStackSize size</td>
<td></td>
</tr>
<tr>
<td>TimeOut seconds</td>
<td>300</td>
</tr>
<tr>
<td>TraceEnable on</td>
<td>off</td>
</tr>
<tr>
<td>TRACE</td>
<td></td>
</tr>
<tr>
<td>Configuration Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TransferLog file</td>
<td>pipe</td>
</tr>
<tr>
<td>TypesConfig file-path</td>
<td>mime.types</td>
</tr>
<tr>
<td>UnsetEnv env-variable [env-variable] ...</td>
<td></td>
</tr>
<tr>
<td>UseCanonicalName On</td>
<td>Off</td>
</tr>
<tr>
<td>UseCanonicalPhysicalPort On</td>
<td>Off</td>
</tr>
<tr>
<td>User unix-userid</td>
<td>ID</td>
</tr>
<tr>
<td>UserDir directory-filename</td>
<td></td>
</tr>
<tr>
<td>VirtualDocumentRoot interpolated-directory</td>
<td>none</td>
</tr>
<tr>
<td>VirtualDocumentRootIP interpolated-directory</td>
<td>none</td>
</tr>
<tr>
<td>&lt;VirtualHost addr[:port] [addr[:port]] ...&gt; ...</td>
<td></td>
</tr>
<tr>
<td>&lt;/VirtualHost&gt;</td>
<td></td>
</tr>
<tr>
<td>VirtualScriptAlias interpolated-directory</td>
<td>none</td>
</tr>
<tr>
<td>VirtualScriptAliasIP interpolated-directory</td>
<td>none</td>
</tr>
<tr>
<td>Win32DisableAcceptEx accept() AcceptEx</td>
<td></td>
</tr>
<tr>
<td>XBitHack on</td>
<td>off</td>
</tr>
<tr>
<td>SSI</td>
<td></td>
</tr>
</tbody>
</table>
Apache

(MPMs)
core
  Apache HTTP

mpm_common
  (MPM)

beos
  This Multi-Processing Module is optimized for BeOS.

event
  An experimental variant of the standard worker MPM

mpm_netware
  Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare

mpmt_os2
  Hybrid multi-process, multi-threaded MPM for OS/2

prefork
  fork

mpm_winnt
  Windows NT

worker
mod_actions
  CGI

mod_alias

mod_asis
  HTTP

mod_auth_basic

mod_auth_digest
  User authentication using MD5 Digest Authentication.

mod_authn_alias
  Provides the ability to create extended authentication providers based on actual providers

mod_authn_anon
  "anonymous"

mod_authn_dbd
  User authentication using an SQL database

mod_authn_dbm
  DBM

mod_authn_default

mod_authn_file

mod_authnz_ldap
  Allows an LDAP directory to be used to store the database for HTTP Basic authentication.
mod_authz_dbm
Group authorization using DBM files

mod_authz_default

mod_authz_groupfile

mod_authz_host
(IP)

mod_authz_owner

mod_authz_user

mod_autoindex
Unix ls Win32 dir

mod_cache
URI

mod_cern_meta
CERN httpd metafile semantics

mod_cgi
CGI

mod_cgid
CGI CGI

mod_charset_lite
Specify character set translation or recoding

mod_dav
(WebDAV)

mod_dav_fs
mod_dav

mod_dav_lock
mod_dav

mod_dbd
  Manages SQL database connections

mod_deflate

mod_dir

mod_disk_cache
  URI

mod_dumpio
  I/O

mod_echo

mod_env
  CGI  SSI

mod_example
  Illustrates the Apache module API

mod_expires
  Expires  Cache-Control  HTTP

mod_ext_filter

mod_file_cache
  Caches a static list of files in memory

mod_filter
  Context-sensitive smart filter configuration module

mod_headers
  HTTP

mod_ident
  RFC 1413 ident lookups
**mod_imagemap**
Server-side imagemap processing

**mod_include**
html  (Server Side Includes)

**mod_info**

**mod_isapi**
ISAPI Extensions within Apache for Windows

**mod_ldap**
LDAP connection pooling and result caching services for use by other LDAP modules

**mod_log_config**

**mod_log_forensic**
forensic

**mod_logio**

**mod_mem_cache**
URI

**mod_mime**

**mod_mime_magic**
Determines the MIME type of a file by looking at a few bytes of its contents

**mod_negotiation**

**mod_nw_ssl**
Enable SSL encryption for NetWare

**mod_proxy**
HTTP/1.1 /

**mod_proxy_ajp**
mod_proxy AJP

**mod_proxy_balancer**
mod_proxy

**mod_proxy_connect**
CONNECT mod_proxy

**mod_proxy_ftp**
mod_proxy FTP

**mod_proxy_http**
mod_proxy HTTP

**mod_proxy_scgi**
mod_proxy SCGI

**mod_reqtimeout**
Set timeout and minimum data rate for receiving requests

**mod_rewrite**
Provides a rule-based rewriting engine to rewrite requested URLs on the fly

**mod_setenvif**

**mod_so**

**mod_speling**
URL

**mod_ssl**
Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols

**mod_status**
mod_substitute
Perform search and replace operations on response bodies

mod_suexec
CGI

mod_unique_id

mod_userdir

mod_usertrack
Clickstream logging of user activity on a site

mod_version

mod_vhost_alias
Provides for dynamically configured mass virtual hosting
Site Map

Apache HTTP 2.2
- 2.0 2.2
- Apache 2.1/2.2
- Apache 2.0
- Apache License
- Apache
- Apache
- Directory, Location, Files
- URL
- (DSO)
- Apache
- (MPM)
- Apache
- Apache
- suEXEC
- URL
- IP
- VirtualHost
- DNS_Apache
• CGI
• Server Side Includes
• .htaccess
- Microsoft Windows Apache
- Microsoft Windows Apache
- Novell NetWare Apache
- HPUX
- EBCDIC Apache
• : httpd
• : ab
• : apachectl
• : apxs
• : configure
• : dbmmanage
• : htcacheclean
• : htdbm
• : htdigest
• : htpassword
• : logresolve
• : rotatelogs
• : suexec
- Apache
- Apache

- Apache
- Apache MPM
- Apache MPM beos
- Apache MPM event
- Apache MPM netware
- Apache MPM os2
- Apache MPM prefork
- Apache MPM winnt
- Apache MPM worker

- Apache_mod_actions
- Apache_mod_alias
- Apache_mod_asis
- Apache_mod_auth_basic
- Apache_mod_auth_digest
- Apache_mod_authn_alias
- Apache_mod_authn_anon
- Apache_mod_authn_dbd
- Apache_mod_authn_dbm
- Apache_mod_authn_default
- Apache_mod_authn_file
- Apache_mod_authnz_ldap
- Apache_mod_authz_dbm
- Apache_mod_authz_default
- Apache_mod_authz_groupfile
- Apache_mod_authz_host
- Apache_mod_authz_owner
- Apache_mod_authz_user
- Apache_mod_autoindex
- Apache_mod_cache
- Apache mod_cern_meta
- Apache mod_cgi
- Apache mod_cgid
- Apache mod_charset_lite
- Apache mod_dav
- Apache mod_dav_fs
- Apache mod_dav_lock
- Apache mod_dbd
- Apache mod_deflate
- Apache mod_dir
- Apache mod_disk_cache
- Apache mod_dumpio
- Apache mod_echo
- Apache mod_env
- Apache mod_example
- Apache mod_expires
- Apache mod_ext_filter
- Apache mod_file_cache
- Apache mod_filter
- Apache mod_headers
- Apache mod_ident
- Apache mod_imagemap
- Apache mod_include
- Apache mod_info
- Apache mod_isapi
- Apache mod_Idap
- Apache mod_log_config
- Apache mod_log_forensic
- Apache mod_logio
- Apache mod_mem_cache
- Apache mod_mime
- Apache mod_mime_magic
- Apache mod_negotiation
- Apache mod_nw_ssl
- Apache mod_proxy
- Apache mod_proxy_ajp
- Apache mod_proxy_balancer
- Apache mod_proxy_connect
- Apache mod_proxy_ftp
- Apache mod_proxy_http
- Apache mod_proxy_scgi
- Apache mod_reqtimeout
- Apache mod_rewrite
- Apache mod_setenvif
- Apache mod_so
- Apache mod_speling
- Apache mod_ssl
- Apache mod_status
- Apache mod_substitute
- Apache mod_suexec
- Apache mod_unique_id
- Apache mod_userdir
- Apache mod_usertrack
- Apache mod_version
- Apache mod_vhost_alias
• 

• Apache API
• APR
• Apache 2.0
• Apache 2.0
• Apache 1.3 Apache 2.0
• Apache 2.0
• Apache 2.0
Apache HTTP
httpd
    Apache
apachectl
    Apache HTTP
ab
    Apache HTTP
apxs
    Apache (APache eXtenSion)
configure

dbmmanage
    Basic  DBM
htcacheclean

htdigest
    Digest
htdocs
    DBM
htpasswd
    Basic
httxt2dbm
    RewriteMap  dbm
logresolve
    Apache  IP
rotatelogs
    Apache
suexec
    Exec (CGI)
Apache SSL/TLS

Apache HTTP  mod_ssl  OpenSSL  Secure Sockets Layer
Transport Layer Security  Ralf S. Engelschall  mod_ssl
How-To
mod_rewrite provides a way to modify incoming URL requests, dynamically, based on regular expression rules. This allows you to map arbitrary URLs onto your internal URL structure in any way you like.

It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests: server variables, environment variables, HTTP headers, time stamps, external database lookups, and various other external programs or handlers, can be used to achieve granular URL matching.

Rewrite rules can operate on the full URLs, including the path-info and query string portions, and may be used in per-server context (httpd.conf), per-virtualhost context (<VirtualHost> blocks), or per-directory context (.htaccess files and <Directory> blocks). The rewritten result can lead to further rules, internal sub-processing, external request redirection, or proxy passthrough, depending on what flags you attach to the rules.

Since mod_rewrite is so powerful, it can indeed be rather complex. This document supplements the reference documentation, and attempts to allay some of that complexity, and provide highly annotated examples of common scenarios that you may handle with mod_rewrite. But we also attempt to show you when you should not use mod_rewrite, and use other standard Apache features instead, thus avoiding this unnecessary complexity.

- mod_rewrite reference documentation
- Introduction to regular expressions and mod_rewrite
- Using mod_rewrite for redirection and remapping of URLs
• Using mod_rewrite to control access
• Dynamic virtual hosts with mod_rewrite
• Dynamic proxying with mod_rewrite
• Using RewriteMap
• Advanced techniques and tricks
• When NOT to use mod_rewrite
• RewriteRule Flags
• Technical details

See also

mod_rewrite reference documentation
Mapping URLs to the Filesystem
mod_rewrite wiki
Glossary
Apache

1 

Apache 1.1 Apache

Apache 1.3

mod_vhost_alias

IP
(IP)
IP (IP)

( )

•

•

•
- <VirtualHost>
- NameVirtualHost
- ServerName
- ServerAlias
- ServerPath

```
Apache -S
```

```
/usr/local/apache2/bin/httpd -S
```

<table>
<thead>
<tr>
<th>Apache</th>
<th>IP</th>
</tr>
</thead>
</table>

Copyright 2017 The Apache Software Foundation.

Apache License, Version 2.0.
Frequently Asked Questions

The FAQ has moved to the HTTP Server Wiki.
Warning

Many of the documents listed here are in need of update. They are in different stages of progress. Please be patient, and point out any discrepancies or errors on the developer/ pages directly to the dev@httpd.apache.org mailing list.
- Apache 1.3 API Notes
- Apache 2.0 Hook Functions
- Request Processing in Apache 2.0
- How filters work in Apache 2.0
- Converting Modules from Apache 1.3 to Apache 2.0
- Documenting Apache 2.0
- Apache 2.x Thread Safety Issues
Developer articles at apachetutor include:
- Request Processing in Apache
- Configuration for Modules
- Resource Management in Apache
- Connection Pooling in Apache
- Introduction to Buckets and Brigades
Apache Miscellaneous Documentation

Below is a list of additional documentation pages that apply to the Apache web server development project.

**Warning**

The documents below have not been fully updated to take into account changes made in the 2.1 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

**Performance Notes - Apache Tuning**

Notes about how to (run-time and compile-time) configure Apache for highest performance. Notes explaining why Apache does some things, and why it doesn't do other things (which make it slower/faster).

**Security Tips**

Some "do"s - and "don't"s - for keeping your Apache web site secure.

**URL Rewriting Guide**

This document supplements the [mod_rewrite reference documentation](https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html). It describes how one can use Apache's [mod_rewrite](https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html) to solve typical URL-based problems webmasters are usually confronted with in practice.

**Relevant Standards**

This document acts as a reference page for most of the relevant standards that Apache follows.
Password Encryption Formats

Discussion of the various ciphers supported by Apache for authentication purposes.
httxt2dbm - Generate dbm files for use with RewriteMap

httxt2dbm is used to generate dbm files from text input, for use in RewriteMap with the dbm map type.

See also

httpd
mod_rewrite
httxt2dbm [ -v ] [ -f DBM_TYPE ] -i SOURCE_TXT -o OUTPUT_DBM
-v
  More verbose output

-f DBM_TYPE
  Specify the DBM type to be used for the output. If not specified, will use the APR Default. Available types are: GDBM for GDBM files, SDBM for SDBM files, DB for berkeley DB files, NDBM for NDBM files, default for the default DBM type.

-i SOURCE_TXT
  Input file from which the dbm is to be created. The file should be formatted with one record per line, of the form: key value. See the documentation for RewriteMap for further details of this file's format and meaning.

-o OUTPUT_DBM
  Name of the output dbm files.
httxt2dbm -i rewritemap.txt -o rewritemap.dbm
httxt2dbm -f SDBM -i rewritemap.txt -o rewritemap.dbm
Using Apache
This document explains how to install, configure and run Apache 2.0 under Microsoft Windows.

See: Using Apache with Microsoft Windows

Compiling Apache
There are many important points before you begin compiling Apache. This document explain them.

See: Compiling Apache for Microsoft Windows
Novell NetWare

This document explains how to install, configure and run Apache 2.0 under Novell NetWare 5.1 and above.

See: Using Apache With Novell NetWare

EBCDIC

Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

Warning: This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

See: The Apache EBCDIC Port
suexec is used by the Apache HTTP Server to switch to another user before executing CGI programs. In order to achieve this, it must run as root. Since the HTTP daemon normally doesn't run as root, the suexec executable needs the setuid bit set and must be owned by root. It should never be writable for any other person than root.

For further information about the concepts and the security model of suexec please refer to the suexec documentation (http://httpd.apache.org/docs/2.2/suexec.html).
suexec -V
-V

If you are root, this option displays the compile options of suexec. For security reasons all configuration options are changeable only at compile time.
How-To /
See:

**CGI**

CGI (Common Gateway Interface)

*: CGI:

**.htaccess**

.htaccess

*: .htaccess

**Server Side Includes**

SSI (Server Side Includes) HTML

HTML

*: Server Side Includes (SSI)

UserDir

http://example.com/~username/

UserDir

*: ( public_html)
Password Formats

Notes about the password encryption formats generated and understood by Apache.
There are four formats that Apache recognizes for basic-authentication passwords. Note that not all formats work on every platform:

**PLAIN TEXT (i.e. unencrypted)**
Windows, BEOS, & Netware only.

**CRYPT**
Unix only. Uses the traditional Unix crypt(3) function with a randomly-generated 32-bit salt (only 12 bits used) and the first 8 characters of the password.

**SHA1**
"{SHA}" + Base64-encoded SHA-1 digest of the password.

**MD5**
"$apr1$" + the result of an Apache-specific algorithm using an iterated (1,000 times) MD5 digest of various combinations of a random 32-bit salt and the password. See the APR source file `apr_md5.c` for the details of the algorithm.

**Generating values with htpasswd**

**MD5**
```
$ htpasswd -nbm myName myPassword
myName:$apr1$r31.....$HqJZimcKQFAMYayBlzkrA/
```

**SHA1**
```
$ htpasswd -nbs myName myPassword
myName:{SHA}VBPuJHI7uixaa6LQGwx4s+5GKNE=
```

**CRYPT**
```
$ htpasswd -nbd myName myPassword
myName:rqXexS6Z hobKA
```
Generating CRYPT and MD5 values with the OpenSSL command-line program

OpenSSL knows the Apache-specific MD5 algorithm.

**MD5**

```
$ openssl passwd -apr1 myPassword
$apr1$qHDFfhPC$nITSVHGyBdAK1Y0acGRnY0
```

**CRYPT**

```
openssl passwd -crypt myPassword
qQ5vTYO3c8dsU
```

Validating CRYPT or MD5 passwords with the OpenSSL command line program

The salt for a CRYPT password is the first two characters (converted to a binary value). To validate myPassword against rqXexS6ZhobKA

**CRYPT**

```
$ openssl passwd -crypt -salt rq myPassword
Warning: truncating password to 8 characters
rqXexS6ZhobKA
```

Note that using myPasswo instead of myPassword will produce the same result because only the first 8 characters of CRYPT passwords are considered.

The salt for an MD5 password is between $apr1$ and the following $ (as a Base64-encoded binary value - max 8 chars). To validate myPassword against $apr1$r31.....$HqJZimcKQFAMYayBlzkrA/

**MD5**

```
```
Database password fields for mod_dbd

The SHA1 variant is probably the most useful format for DBD authentication. Since the SHA1 and Base64 functions are commonly available, other software can populate a database with encrypted passwords that are usable by Apache basic authentication.

To create Apache SHA1-variant basic-authentication passwords in various languages:

**PHP**

'{{SHA}}' . base64_encode(sha1($password, TRUE))

**Java**

"{SHA}" + new sun.misc.BASE64Encoder().encode(java.security.MessageDigest.getInstance("SHA1").digest(password.getBytes()))

**ColdFusion**

"{SHA}" & ToBase64(BinaryDecode(Hash(password, "SHA1"), "Hex"))

**Ruby**

require 'digest/sha1'
require 'base64'
'{SHA}' + Base64.encode64(Digest::SHA1.digest(password))

**C or C++**

Use the APR function: apr_sha1_base64

**PostgreSQL (with the contrib/pgcrypto functions installed)**
'${SHA}'||encode(digest(password,'sha1'),'base64')
Apache recognizes one format for digest-authentication passwords - the MD5 hash of the string user:realm:password as a 32-character string of hexadecimal digits. realm is the Authorization Realm argument to the AuthName directive in httpd.conf.

**Database password fields for mod_dbd**

Since the MD5 function is commonly available, other software can populate a database with encrypted passwords that are usable by Apache digest authentication.

To create Apache digest-authentication passwords in various languages:

**PHP**

```php
md5($user . '::' . $realm . '::' . $password)
```

**Java**

```java
byte b[] =
java.security.MessageDigest.getInstance("MD5").digest( (user + "::" + realm + "::" + password ).getBytes());
java.math.BigInteger bi = new java.math.BigInteger(1, b);
String s = bi.toString(16);
while (s.length() < 32)
    s = "0" + s;
// String s is the encrypted password
```

**ColdFusion**

```coldfusion
LCase(Hash( (user & "::" & realm & "::" & password) , "MD5"))
```

**Ruby**

```ruby
require 'digest/md5'
Digest::MD5.hexdigest(user + '::' + realm + '::' + password)
```
PostgreSQL (with the contrib/pgcrypto functions installed)

```sql
encode(digest(user || ':' || realm || ':' || password, 'md5'), 'hex')
```
Apache mod_rewite Introduction

This document supplements the mod_rewite reference documentation. It describes the basic concepts necessary for use of mod_rewite. Other documents go into greater detail, but this doc should help the beginner get their feet wet.

See also

- Module documentation
- Redirection and remapping
- Controlling access
- Virtual hosts
- Proxying
- Using RewriteMap
- Advanced techniques
- When not to use mod_rewite
The Apache module `mod_rewrite` is a very powerful and sophisticated module which provides a way to do URL manipulations. With it, you can do nearly all types of URL rewriting that you may need. It is, however, somewhat complex, and may be intimidating to the beginner. There is also a tendency to treat rewrite rules as magic incantation, using them without actually understanding what they do.

This document attempts to give sufficient background so that what follows is understood, rather than just copied blindly.

Remember that many common URL-manipulation tasks don't require the full power and complexity of `mod_rewrite`. For simple tasks, see `mod_alias` and the documentation on mapping URLs to the filesystem.

Finally, before proceeding, be sure to configure `mod_rewrite`'s log level to one of the trace levels using the `LogLevel` directive. Although this can give an overwhelming amount of information, it is indispensable in debugging problems with `mod_rewrite` configuration, since it will tell you exactly how each rule is processed.
mod_rewrite uses the Perl Compatible Regular Expression vocabulary. In this document, we do not attempt to provide a detailed reference to regular expressions. For that, we recommend the PCRE man pages, the Perl regular expression man page, and Mastering Regular Expressions, by Jeffrey Friedl.

In this document, we attempt to provide enough of a regex vocabulary to get you started, without being overwhelming, in the hope that RewriteRules will be scientific formulae, rather than magical incantations.

**Regex vocabulary**

The following are the minimal building blocks you will need, in order to write regular expressions and RewriteRules. They certainly do not represent a complete regular expression vocabulary, but they are a good place to start, and should help you read basic regular expressions, as well as write your own.

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Matches any single character</td>
<td>c . t will match cat, cot, cut, etc.</td>
</tr>
<tr>
<td>+</td>
<td>Repeats the previous match one or more times</td>
<td>a+ matches a, aa, aaa, etc</td>
</tr>
<tr>
<td>*</td>
<td>Repeats the previous match zero or more times.</td>
<td>a * matches all the same things a+ matches, but will also match an empty string.</td>
</tr>
<tr>
<td>?</td>
<td>Makes the match optional.</td>
<td>colou?r will match color and colour.</td>
</tr>
<tr>
<td>^</td>
<td>Called an anchor, matches the beginning of the string</td>
<td>^a matches a string that begins with a</td>
</tr>
</tbody>
</table>
$ The other anchor, this matches the end of the string. a$ matches a string that ends with a.

( ) Groups several characters into a single unit, and captures a match for use in a backreference. (ab)+ matches ababab - that is, the + applies to the group. For more on backreferences see below.

[ ] A character class - matches one of the characters c[ua]t matches cut, cot or cat.

[^ ] Negative character class - matches any character not specified c[^a]t matches cat or c=t but not c/t

In mod_rewrite the ! character can be used before a regular expression to negate it. This is, a string will be considered to have matched only if it does not match the rest of the expression.

**Regex Back-Reference Availability**

One important thing here has to be remembered: Whenever you use parentheses in Pattern or in one of the CondPattern, back-references are internally created which can be used with the strings $N and %N (see below). These are available for creating the strings Substitution and TestString as outlined in the following chapters. Figure 1 shows to which locations the back-references are transferred for expansion as well as illustrating the flow of the RewriteRule, RewriteCond matching. In the next chapters, we will be exploring how to use these back-references, so do not fret if it seems a bit alien to you at first.
Figure 1: The back-reference flow through a rule.
In this example, a request for /test/1234 would be transformed into /admin.foo?
page=test&id=1234&host=admin.example.com.
A **RewriteRule** consists of three arguments separated by spaces. The arguments are

1. **Pattern**: which incoming URLs should be affected by the rule;
2. **Substitution**: where should the matching requests be sent;
3. **[flags]**: options affecting the rewritten request.

The **Pattern** is a regular expression. It is initially (for the first rewrite rule or until a substitution occurs) matched against the URL-path of the incoming request (the part after the hostname but before any question mark indicating the beginning of a query string) or, in per-directory context, against the request's path relative to the directory for which the rule is defined. Once a substitution has occurred, the rules that follow are matched against the substituted value.

![Syntax of a RewriteRule](image.png)

**Figure 2**: Syntax of the RewriteRule directive.

The **Substitution** can itself be one of three things:

A full filesystem path to a resource
This maps a request to an arbitrary location on your filesystem, much like the `Alias` directive.

**A web-path to a resource**

```
RewriteRule ^/foo$ /bar
```

If `DocumentRoot` is set to `/usr/local/apache2/htdocs`, then this directive would map requests for `http://example.com/foo` to the path `/usr/local/apache2/htdocs/bar`.

**An absolute URL**

```
RewriteRule ^/product/view$ http://site2.example.com/seeproduct.html [R]
```

This tells the client to make a new request for the specified URL.

The `Substitution` can also contain `back-references` to parts of the incoming URL-path matched by the `Pattern`. Consider the following:

```
RewriteRule ^/product/(.*)/view$ /var/web/productdb/$1
```

The variable `$1` will be replaced with whatever text was matched by the expression inside the parenthesis in the `Pattern`. For example, a request for `http://example.com/product/r14df/view` will be mapped to the path `/var/web/productdb/r14df`.

If there is more than one expression in parenthesis, they are
available in order in the variables $1$, $2$, $3$, and so on.
The behavior of a RewriteRule can be modified by the application of one or more flags to the end of the rule. For example, the matching behavior of a rule can be made case-insensitive by the application of the [NC] flag:

```
RewriteRule ^puppy.html smalldog.html [NC]
```

For more details on the available flags, their meanings, and examples, see the Rewrite Flags document.
One or more RewriteCond directives can be used to restrict the types of requests that will be subject to the following RewriteRule. The first argument is a variable describing a characteristic of the request, the second argument is a regular expression that must match the variable, and a third optional argument is a list of flags that modify how the match is evaluated.

![Syntax of a RewriteCond](image)

**Figure 3: Syntax of the RewriteCond directive**

For example, to send all requests from a particular IP range to a different server, you could use:

```
RewriteCond %{REMOTE_ADDR} ^10\.2\.
RewriteRule (\.*) http://intranet.example.com$1
```

When more than one RewriteCond is specified, they must all match for the RewriteRule to be applied. For example, to deny requests that contain the word "hack" in their query string, unless they also contain a cookie containing the word "go", you could use:

```
RewriteCond %{QUERY_STRING} hack
RewriteCond %{HTTP_COOKIE} !go
```
Notice that the exclamation mark specifies a negative match, so the rule is only applied if the cookie does not contain "go".

Matches in the regular expressions contained in the `RewriteCond` can be used as part of the `Substitution` in the `RewriteRule` using the variables %1, %2, etc. For example, this will direct the request to a different directory depending on the hostname used to access the site:

```
RewriteCond %{HTTP_HOST} (.*
RewriteRule ^/(.*) /sites/%1/$1
```

If the request was for http://example.com/foo/bar, then %1 would contain example.com and $1 would contain foo/bar.
The **RewriteMap** directive provides a way to call an external function, so to speak, to do your rewriting for you. This is discussed in greater detail in the [RewriteMap supplementary documentation](#).
Rewriting is typically configured in the main server configuration setting (outside any `<Directory>` section) or inside `<VirtualHost>` containers. This is the easiest way to do rewriting and is recommended. It is possible, however, to do rewriting inside `<Directory>` sections or `.htaccess` files at the expense of some additional complexity. This technique is called per-directory rewrites.

The main difference with per-server rewrites is that the path prefix of the directory containing the `.htaccess` file is stripped before matching in the `RewriteRule`. In addition, the `RewriteBase` should be used to assure the request is properly mapped.
RewriteRule Flags

This document discusses the flags which are available to the RewriteRule directive, providing detailed explanations and examples.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Virtual hosts
- Proxying
- Using RewriteMap
- Advanced techniques
- When not to use mod_rewrite
A **RewriteRule** can have its behavior modified by one or more flags. Flags are included in square brackets at the end of the rule, and multiple flags are separated by commas.

```
RewriteRule pattern target [Flag1,Flag2,Flag3]
```

Each flag (with a few exceptions) has a short form, such as C0, as well as a longer form, such as cookie. While it is most common to use the short form, it is recommended that you familiarize yourself with the long form, so that you remember what each flag is supposed to do. Some flags take one or more arguments. Flags are not case sensitive.

Flags that alter metadata associated with the request (T=, H=, E=) have no affect in per-directory and .htaccess context, when a substitution (other than `-`) is performed during the same round of rewrite processing.

Presented here are each of the available flags, along with an example of how you might use them.
The [B] flag instructs `RewriteRule` to escape non-alphanumeric characters before applying the transformation.

`mod_rewrite` has to unescape URLs before mapping them, so backreferences will be unescaped at the time they are applied. Using the B flag, non-alphanumeric characters in backreferences will be escaped. For example, consider the rule:

```
RewriteRule ^search/(.*)$ /search.php?term=$1
```

Given a search term of 'x & y/z', a browser will encode it as 'x%20%26%20y%2Fz', making the request 'search/x%20%26%20y%2Fz'. Without the B flag, this rewrite rule will map to 'search.php?term=x & y/z', which isn't a valid URL, and so would be encoded as search.php?term=x%20&y%2Fz=, which is not what was intended.

With the B flag set on this same rule, the parameters are re-encoded before being passed on to the output URL, resulting in a correct mapping to /search.php?term=x%20%26%20y%2Fz.

Note that you may also need to set `AllowEncodedSlashes` to On to get this particular example to work, as httpd does not allow encoded slashes in URLs, and returns a 404 if it sees one.

This escaping is particularly necessary in a proxy situation, when the backend may break if presented with an unescaped URL.
The [C] or [chain] flag indicates that the RewriteRule is chained to the next rule. That is, if the rule matches, then it is processed as usual and control moves on to the next rule. However, if it does not match, then the next rule, and any other rules that are chained together, will be skipped.
The [CO], or [cookie] flag, allows you to set a cookie when a particular RewriteRule matches. The argument consists of three required fields and four optional fields.

The full syntax for the flag, including all attributes, is as follows:

```
```

You must declare a name, a value, and a domain for the cookie to be set.

**Domain**

The domain for which you want the cookie to be valid. This may be a hostname, such as `www.example.com`, or it may be a domain, such as `.example.com`. It must be at least two parts separated by a dot. That is, it may not be merely `.com` or `.net`. Cookies of that kind are forbidden by the cookie security model.

You may optionally also set the following values:

**Lifetime**

The time for which the cookie will persist, in minutes. A value of 0 indicates that the cookie will persist only for the current browser session. This is the default value if none is specified. Prior to 2.2.28, a value of 0 indicates immediate expiration and it was not possible to specify session lifetime if any later parameters (Path, Secure, httponly) were specified.

**Path**

The path, on the current website, for which the cookie is valid, such as `/customers/` or `/files/download/`.

By default, this is set to `/` - that is, the entire website.

**Secure**
If set to secure, true, or 1, the cookie will only be permitted to be translated via secure (https) connections.

**HttpOnly**
If set to HttpOnly, true, or 1, the cookie will have the HttpOnly flag set, which means that the cookie will be inaccessible to JavaScript code on browsers that support this feature.

Several examples are offered here:

```plaintext
RewriteEngine On
RewriteRule ^/index\.html - 
[CO=frontdoor:yes:.example.com:1440:/]
```

In the example given, the rule doesn't rewrite the request. The "-" rewrite target tells mod_rewrite to pass the request through unchanged. Instead, it sets a cookie called 'frontdoor' to a value of 'yes'. The cookie is valid for any host in the .example.com domain. It will be set to expire in 1440 minutes (24 hours) and will be returned for all URIs.
The DPI flag causes the PATH_INFO portion of the rewritten URI to be discarded.

This flag is available in version 2.2.12 and later.

In per-directory context, the URI each RewriteRule compares against is the concatenation of the current values of the URI and PATH_INFO.

The current URI can be the initial URI as requested by the client, the result of a previous round of mod_rewrite processing, or the result of a prior rule in the current round of mod_rewrite processing.

In contrast, the PATH_INFO that is appended to the URI before each rule reflects only the value of PATH_INFO before this round of mod_rewrite processing. As a consequence, if large portions of the URI are matched and copied into a substitution in multiple RewriteRule directives, without regard for which parts of the URI came from the current PATH_INFO, the final URI may have multiple copies of PATH_INFO appended to it.

Use this flag on any substitution where the PATH_INFO that resulted from the previous mapping of this request to the filesystem is not of interest. This flag permanently forgets the PATH_INFO established before this round of mod_rewrite processing began. PATH_INFO will not be recalculated until the current round of mod_rewrite processing completes. Subsequent rules during this round of processing will see only the direct result of substitutions, without any PATH_INFO appended.
With the [E], or [env] flag, you can set the value of an environment variable. Note that some environment variables may be set after the rule is run, thus unsetting what you have set. See the Environment Variables document for more details on how Environment variables work.

The full syntax for this flag is:

```
[E=VAR:VAL] [E=!VAR]
```

VAL may contain backreferences ($N or %N) which will be expanded.

Using the short form

```
[E=VAR]
```

you can set the environment variable named VAR to an empty value.

The form

```
[E=!VAR]
```

allows to unset a previously set environment variable named VAR.

Environment variables can then be used in a variety of contexts, including CGI programs, other RewriteRule directives, or CustomLog directives.

The following example sets an environment variable called 'image' to a value of '1' if the requested URI is an image file. Then, that environment variable is used to exclude those requests from the access log.
Note that this same effect can be obtained using SetEnvIf. This technique is offered as an example, not as a recommendation.
Using the [F] flag causes the server to return a 403 Forbidden status code to the client. While the same behavior can be accomplished using the Deny directive, this allows more flexibility in assigning a Forbidden status.

The following rule will forbid .exe files from being downloaded from your server.

```
RewriteRule \.exe - [F]
```

This example uses the "-" syntax for the rewrite target, which means that the requested URI is not modified. There's no reason to rewrite to another URI, if you're going to forbid the request.

When using [F], an [L] is implied - that is, the response is returned immediately, and no further rules are evaluated.
The [G] flag forces the server to return a 410 Gone status with the response. This indicates that a resource used to be available, but is no longer available.

As with the [F] flag, you will typically use the "-" syntax for the rewrite target when using the [G] flag:

```
RewriteRule oldproduct - [G,NC]
```

When using [G], an [L] is implied - that is, the response is returned immediately, and no further rules are evaluated.
Forces the resulting request to be handled with the specified handler. For example, one might use this to force all files without a file extension to be parsed by the php handler:

```
RewriteRule !\.- [H=application/x-httpd-php]
```

The regular expression above - `!\.-` - will match any request that does not contain the literal `. `character.

This can be also used to force the handler based on some conditions. For example, the following snippet used in per-server context allows `.php` files to be *displayed* by `mod_php` if they are requested with the `.phps` extension:

```
RewriteRule ^(/source/.+\.php)s$ $1 [H=application/x-httpd-php-source]
```

The regular expression above - `^(/source/.+\.php)s$` - will match any request that starts with `/source/` followed by 1 or n characters followed by `.phps` literally. The backreference `$1` refers to the captured match within parenthesis of the regular expression.
The [L] flag causes `mod_rewrite` to stop processing the rule set. In most contexts, this means that if the rule matches, no further rules will be processed. This corresponds to the `last` command in Perl, or the `break` command in C. Use this flag to indicate that the current rule should be applied immediately without considering further rules.

If you are using `RewriteRule` in either `.htaccess` files or in `<Directory>` sections, it is important to have some understanding of how the rules are processed. The simplified form of this is that once the rules have been processed, the rewritten request is handed back to the URL parsing engine to do what it may with it. It is possible that as the rewritten request is handled, the `.htaccess` file or `<Directory>` section may be encountered again, and thus the ruleset may be run again from the start. Most commonly this will happen if one of the rules causes a redirect - either internal or external - causing the request process to start over.

It is therefore important, if you are using `RewriteRule` directives in one of these contexts, that you take explicit steps to avoid rules looping, and not count solely on the [L] flag to terminate execution of a series of rules, as shown below.

The example given here will rewrite any request to `index.php`, giving the original request as a query string argument to `index.php`, however, the `RewriteCond` ensures that if the request is already for `index.php`, the `RewriteRule` will be skipped.

```bash
RewriteBase /
RewriteCond %{REQUEST_URI} !=/index.php
RewriteRule ^(.*) /index.php?req=$1 [L,PT]
```
The [N] flag causes the ruleset to start over again from the top, using the result of the ruleset so far as a starting point. Use with extreme caution, as it may result in loop.

The [Next] flag could be used, for example, if you wished to replace a certain string or letter repeatedly in a request. The example shown here will replace A with B everywhere in a request, and will continue doing so until there are no more As to be replaced.

```
RewriteRule (.*)A(.*) $1B$2 [N]
```

You can think of this as a while loop: While this pattern still matches (i.e., while the URI still contains an A), perform this substitution (i.e., replace the A with a B).
Use of the [NC] flag causes the `RewriteRule` to be matched in a case-insensitive manner. That is, it doesn't care whether letters appear as upper-case or lower-case in the matched URI.

In the example below, any request for an image file will be proxied to your dedicated image server. The match is case-insensitive, so that `.jpg` and `.JPG` files are both acceptable, for example.

```
RewriteRule .*\.(jpg|gif|png)$ http://images.example.com$1 [P,NC]
```
By default, special characters, such as & and ?, for example, will be converted to their hexcode equivalent. Using the [NE] flag prevents that from happening.

```
RewriteRule ^/anchor/(.+)$ bigpage.html#$1 [NE,R]
```

The above example will redirect /anchor/xyz to /bigpage.html#xyz. Omitting the [NE] will result in the # being converted to its hexcode equivalent, %23, which will then result in a 404 Not Found error condition.
Use of the [NS] flag prevents the rule from being used on subrequests. For example, a page which is included using an SSI (Server Side Include) is a subrequest, and you may want to avoid rewrites happening on those subrequests. Also, when mod_dir tries to find out information about possible directory default files (such as index.html files), this is an internal subrequest, and you often want to avoid rewrites on such subrequests. On subrequests, it is not always useful, and can even cause errors, if the complete set of rules are applied. Use this flag to exclude problematic rules.

To decide whether or not to use this rule: if you prefix URLs with CGI-scripts, to force them to be processed by the CGI-script, it’s likely that you will run into problems (or significant overhead) on sub-requests. In these cases, use this flag.

Images, javascript files, or css files, loaded as part of an HTML page, are not subrequests - the browser requests them as separate HTTP requests.
Use of the [P] flag causes the request to be handled by `mod_proxy`, and handled via a proxy request. For example, if you wanted all image requests to be handled by a back-end image server, you might do something like the following:

```
RewriteRule /(.*)\.\.(jpg|gif|png) http://images.example.com/$1.$2 [P]
```

Use of the [P] flag implies [L] - that is, the request is immediately pushed through the proxy, and any following rules will not be considered.

You must make sure that the substitution string is a valid URI (typically starting with `http://hostname`) which can be handled by the `mod_proxy`. If not, you will get an error from the proxy module. Use this flag to achieve a more powerful implementation of the `ProxyPass` directive, to map remote content into the namespace of the local server.

**Security Warning**

Take care when constructing the target URL of the rule, considering the security impact from allowing the client influence over the set of URLs to which your server will act as a proxy. Ensure that the scheme and hostname part of the URL is either fixed, or does not allow the client undue influence.

Note: `mod_proxy` must be enabled in order to use this flag.
The target (or substitution string) in a RewriteRule is assumed to be a file path, by default. That is to say, the use of the [PT] flag causes the result of the RewriteRule to be passed back through URL mapping, so that location-based mappings, such as Alias, Redirect, or ScriptAlias, for example, might have a chance to take effect.

If, for example, you have an Alias for /icons, and have a RewriteRule pointing there, you should use the [PT] flag to ensure that the Alias is evaluated.

```bash
Alias /icons /usr/local/apache/icons
RewriteRule /pics/(.+).jpg /icons/$1.gif [PT]
```

Omission of the [PT] flag in this case will cause the Alias to be ignored, resulting in a 'File not found' error being returned.

The PT flag implies the L flag: rewriting will be stopped in order to pass the request to the next phase of processing.

Note that the PT flag is implied in per-directory contexts such as `<Directory>` sections or in .htaccess files. The only way to circumvent that is to rewrite to -.
When the replacement URI contains a query string, the default behavior of RewriteRule is to discard the existing query string, and replace it with the newly generated one. Using the [QSA] flag causes the query strings to be combined.

Consider the following rule:

```plaintext
RewriteRule /pages/(.+)/page.php?page=$1 [QSA]
```

With the [QSA] flag, a request for /pages/123?one=two will be mapped to /page.php?page=123&one=two. Without the [QSA] flag, that same request will be mapped to /page.php?page=123 - that is, the existing query string will be discarded.
Use of the [R] flag causes a HTTP redirect to be issued to the browser. If a fully-qualified URL is specified (that is, including http://servername/) then a redirect will be issued to that location. Otherwise, the current protocol, servername, and port number will be used to generate the URL sent with the redirect.

*Any* valid HTTP response status code may be specified, using the syntax [R=305], with a 302 status code being used by default if none is specified. The status code specified need not necessarily be a redirect (3xx) status code. However, if a status code is outside the redirect range (300-399) then the substitution string is dropped entirely, and rewriting is stopped as if the L were used.

In addition to response status codes, you may also specify redirect status using their symbolic names: temp (default), permanent, or seeother.

You will almost always want to use [R] in conjunction with [L] (that is, use [R,L]) because on its own, the [R] flag prepends http://thishost[:thisport] to the URL, but then passes this on to the next rule in the ruleset, which can often result in 'Invalid URI in request' warnings.
The [S] flag is used to skip rules that you don't want to run. The syntax of the skip flag is [S=N], where \( N \) signifies the number of rules to skip (provided the `RewriteRule` matches). This can be thought of as a goto statement in your rewrite ruleset. In the following example, we only want to run the `RewriteRule` if the requested URI doesn't correspond with an actual file.

```
# Is the request for a non-existent file?
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# If so, skip these two RewriteRules
RewriteRule .? - [S=2]
RewriteRule (.*\.gif) images.php?$1
RewriteRule (.*\.html) docs.php?$1
```

This technique is useful because a `RewriteCond` only applies to the `RewriteRule` immediately following it. Thus, if you want to make a `RewriteCond` apply to several `RewriteRules`, one possible technique is to negate those conditions and add a `RewriteRule` with a [Skip] flag. You can use this to make pseudo if-then-else constructs: The last rule of the then-clause becomes skip=N, where \( N \) is the number of rules in the else-clause:

```
# Does the file exist?
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# Create an if-then-else construct by skipping 3 lines if we meant to go to the "else" stanza.
RewriteRule .? - [S=3]

# IF the file exists, then:
   RewriteRule (.*\.gif) images.php?$1
   RewriteRule (.*\.html) docs.php?$1
# Skip past the "else" stanza.
   RewriteRule .? - [S=1]
# ELSE...
   RewriteRule (.* ) 404.php?file=$1
# END
```
Sets the MIME type with which the resulting response will be sent. This has the same effect as the `AddType` directive.

For example, you might use the following technique to serve Perl source code as plain text, if requested in a particular way:

```bash
# Serve .pl files as plain text
RewriteRule \.pl$ - [T=text/plain]
```

Or, perhaps, if you have a camera that produces jpeg images without file extensions, you could force those images to be served with the correct MIME type by virtue of their file names:

```bash
# Files with 'IMG' in the name are jpg images.
RewriteRule IMG - [T=image/jpg]
```

Please note that this is a trivial example, and could be better done using `<FilesMatch>` instead. Always consider the alternate solutions to a problem before resorting to rewrite, which will invariably be a less efficient solution than the alternatives.

If used in per-directory context, use only - (dash) as the substitution `for the entire round of mod_rewrite processing`, otherwise the MIME-type set with this flag is lost due to an internal re-processing (including subsequent rounds of mod_rewrite processing). The L flag can be useful in this context to end the `current round of mod_rewrite processing`. 
Redirecting and Remapping with mod_rewrite

This document supplements the mod_rewrite reference documentation. It describes how you can use mod_rewrite to redirect and remap request. This includes many examples of common uses of mod_rewrite, including detailed descriptions of how each works.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.

See also

- Module documentation
- mod_rewrite introduction
- Controlling access
- Virtual hosts
- Proxying
- Using RewriteMap
- Advanced techniques and tricks
- When not to use mod_rewrite
Description:
Assume we have recently renamed the page foo.html to bar.html and now want to provide the old URL for backward compatibility. However, we want that users of the old URL even not recognize that the pages was renamed - that is, we don't want the address to change in their browser.

Solution:
We rewrite the old URL to the new one internally via the following rule:

```
RewriteEngine on
RewriteRule ^/foo\.html$ /bar.html [PT]
```
Description:
Assume again that we have recently renamed the page foo.html to bar.html and now want to provide the old URL for backward compatibility. But this time we want that the users of the old URL get hinted to the new one, i.e. their browsers Location field should change, too.

Solution:
We force a HTTP redirect to the new URL which leads to a change of the browsers and thus the users view:

```plaintext
RewriteEngine on
RewriteRule ^/foo\.html$ bar.html [R]
```

Discussion
In this example, as contrasted to the internal example above, we can simply use the Redirect directive. mod_rewrite was used in that earlier example in order to hide the redirect from the client:

```plaintext
Redirect /foo.html /bar.html
```
Description:
If a resource has moved to another server, you may wish to have URLs continue to work for a time on the old server while people update their bookmarks.

Solution:
You can use `mod_rewrite` to redirect these URLs to the new server, but you might also consider using the Redirect or RedirectMatch directive.

With mod_rewrite
RewriteEngine on
RewriteRule ^/docs/(.+)$ http://new.example.com/docs/$1 [R,L]

With RedirectMatch
RedirectMatch ^/docs/(.*) http://new.example.com/docs/$1

With Redirect
Redirect /docs/ http://new.example.com/docs/
**Description:**
How can we transform a static page foo.html into a dynamic variant foo.cgi in a seamless way, i.e. without notice by the browser/user.

**Solution:**
We just rewrite the URL to the CGI-script and force the handler to be **cgi-script** so that it is executed as a CGI program. This way a request to /~quux/foo.html internally leads to the invocation of /~quux/foo.cgi.

```plaintext
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ foo.cgi [H=cgi-script]
```
Description:
How can we make URLs backward compatible (still existing virtually) after migrating document.YYYY to document.XXXX, e.g. after translating a bunch of .html files to .php?

Solution:
We rewrite the name to its basename and test for existence of the new extension. If it exists, we take that name, else we rewrite the URL to its original state.

```bash
# backward compatibility ruleset for
# rewriting document.html to document.php
# when and only when document.php exists
<Directory /var/www/htdocs>
  RewriteEngine on
  RewriteBase /var/www/htdocs

  RewriteCond $1.php -f
  RewriteCond $1.html !-f
  RewriteRule ^\1.html$ $1.php
</Directory>
```

Discussion
This example uses an often-overlooked feature of mod_rewrite, by taking advantage of the order of execution of the ruleset. In particular, mod_rewrite evaluates the left-hand-side of the RewriteRule before it evaluates the RewriteCond directives. Consequently, $1 is already defined by the time the RewriteCond directives are evaluated. This allows us to test for the existence of the original (document.html) and target (document.php) files using the same base filename.

This ruleset is designed to use in a per-directory context (In a <Directory> block or in a .htaccess file), so that the -f checks are looking at the correct directory path. You may need to set
a `RewriteBase` directive to specify the directory base that you're working in.
Description:
The goal of this rule is to force the use of a particular hostname, in preference to other hostnames which may be used to reach the same site. For example, if you wish to force the use of www.example.com instead of example.com, you might use a variant of the following recipe.

Solution:
The very best way to solve this doesn't involve mod_rewrite at all, but rather uses the Redirect directive placed in a virtual host for the non-canonical hostname(s).

```xml
<VirtualHost *:80>
    ServerName undesired.example.com
    ServerAlias example.com notthis.example.com
    Redirect / http://www.example.com/
</VirtualHost>

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

If, for whatever reason, you still want to use mod_rewrite - if, for example, you need this to work with a larger set of RewriteRules - you might use one of the recipes below.

For sites running on a port other than 80:

```apache
RewriteCond %{HTTP_HOST} !^www\..example\..com [NC]
RewriteCond %{HTTP_HOST} !^$
RewriteCond %{SERVER_PORT} !^80$
RewriteRule ^/?(.*) http://www.example.com:%{SERVER_PORT}/$1 [L,R,NE]
```

And for a site running on port 80:

```apache
RewriteCond %{HTTP_HOST} !^www\..example\..com [NC]
```
<table>
<thead>
<tr>
<th>Rule Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>RewriteCond %{HTTP_HOST} !^$</td>
</tr>
<tr>
<td>RewriteRule ^/?(.*) <a href="http://www.example.com/$1">http://www.example.com/$1</a> [L,R,NE]</td>
</tr>
</tbody>
</table>

If you wanted to do this generically for all domain names - that is, if you want to redirect **example.com** to **www.example.com** for all possible values of **example.com**, you could use the following recipe:

<table>
<thead>
<tr>
<th>Rule Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>RewriteCond %{HTTP_HOST} !^www.. [NC]</td>
</tr>
<tr>
<td>RewriteCond %{HTTP_HOST} !^$</td>
</tr>
<tr>
<td>RewriteRule ^/?(.*) <a href="http://www.%25%7BHTTP_HOST%7D/$1">http://www.%{HTTP_HOST}/$1</a> [L,R,NE]</td>
</tr>
</tbody>
</table>

These rulesets will work either in your main server configuration file, or in a `.htaccess` file placed in the `DocumentRoot` of the server.
Description:
A particular resource might exist in one of several places, and we want to look in those places for the resource when it is requested. Perhaps we've recently rearranged our directory structure, dividing content into several locations.

Solution:
The following ruleset searches in two directories to find the resource, and, if not finding it in either place, will attempt to just serve it out of the location requested.

```
RewriteEngine on

# first try to find it in dir1/...
# ...and if found stop and be happy:
RewriteCond %{DOCUMENT_ROOT}/dir1/%{REQUEST_URI} -f
RewriteRule ^(.+) %{DOCUMENT_ROOT}/dir1/$1 [L]

# second try to find it in dir2/...
# ...and if found stop and be happy:
RewriteCond %{DOCUMENT_ROOT}/dir2/%{REQUEST_URI} -f
RewriteRule ^(.+) %{DOCUMENT_ROOT}/dir2/$1 [L]

# else go on for other Alias or ScriptAlias directives, # etc.
RewriteRule ^ - [PT]
```
Description:
We have numerous mirrors of our website, and want to redirect people to the one that is located in the country where they are located.

Solution:
Looking at the hostname of the requesting client, we determine which country they are coming from. If we can't do a lookup on their IP address, we fall back to a default server.

We'll use a RewriteMap directive to build a list of servers that we wish to use.

```
HostnameLookups on
RewriteEngine on
RewriteMap multiplex txt:/path/to/map.mirrors
RewriteCond %{REMOTE_HOST} ([a-z]+)$ [NC]
RewriteRule ^/(.*)$ ${multiplex:%1|http://www.example.com/}$1 [R,L]
```

```
## map.mirrors -- Multiplexing Map

de http://www.example.de/
uk http://www.example.uk/
com http://www.example.com/
##EOF##
```

Discussion

This ruleset relies on HostNameLookups being set on, which can be a significant performance hit.

The RewriteCond directive captures the last portion of the hostname of the requesting client - the country code - and the following RewriteRule uses that value to look up the appropriate mirror host in the map file.
Description:
We wish to provide different content based on the browser, or user-agent, which is requesting the content.

Solution:
We have to decide, based on the HTTP header "User-Agent", which content to serve. The following config does the following: If the HTTP header "User-Agent" contains "Mozilla/3", the page foo.html is rewritten to foo.NS.html and the rewriting stops. If the browser is "Lynx" or "Mozilla" of version 1 or 2, the URL becomes foo.20.html. All other browsers receive page foo.32.html. This is done with the following ruleset:

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

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

RewriteRule ^foo\.html$ foo.32.html [L]
```
Description:
On some webservers there is more than one URL for a resource. Usually there are canonical URLs (which are be actually used and distributed) and those which are just shortcuts, internal ones, and so on. Independent of which URL the user supplied with the request, they should finally see the canonical one in their browser address bar.

Solution:
We do an external HTTP redirect for all non-canonical URLs to fix them in the location view of the Browser and for all subsequent requests. In the example ruleset below we replace /puppies and /canines by the canonical /dogs.

```conf
RewriteRule ^/(puppies|canines)/(.* )/dogs/$2 [R]
```

Discussion:
This should really be accomplished with Redirect or RedirectMatch directives:

```conf
RedirectMatch ^/(puppies|canines)/(.* )/dogs/$2
```
**Description:**

Usually the `DocumentRoot` of the webserver directly relates to the URL "/". But often this data is not really of top-level priority. For example, you may wish for visitors, on first entering a site, to go to a particular subdirectory `/about/`. This may be accomplished using the following ruleset:

**Solution:**

We redirect the URL / to `/about/`:

```plaintext
RewriteEngine on
RewriteRule ^/$ /about/ [R]
```

Note that this can also be handled using the `RedirectMatch` directive:

```plaintext
RedirectMatch ^/$ http://example.com/about/
```

Note also that the example rewrites only the root URL. That is, it rewrites a request for `http://example.com/`, but not a request for `http://example.com/page.html`. If you have in fact changed your document root - that is, if **all** of your content is in fact in that subdirectory, it is greatly preferable to simply change your `DocumentRoot` directive, or move all of the content up one directory, rather than rewriting URLs.
Description:
You want a single resource (say, a certain file, like index.php) to handle all requests that come to a particular directory, except those that should go to an existing resource such as an image, or a css file.

Solution:
As of version 2.2.16, you should use the FallbackResource directive for this:

```html
<Directory /var/www/my_blog>
  FallbackResource index.php
</Directory>
```

However, in earlier versions of Apache, or if your needs are more complicated than this, you can use a variation of the following rewrite set to accomplish the same thing:

```html
<Directory /var/www/my_blog>
  RewriteBase /my_blog
  RewriteCond /var/www/my_blog/%{REQUEST_FILENAME} !-f
  RewriteCond /var/www/my_blog/%{REQUEST_FILENAME} !-d
  RewriteRule ^ index.php [PT]
</Directory>
```

If, on the other hand, you wish to pass the requested URI as a query string argument to index.php, you can replace that RewriteRule with:

```html
RewriteRule (.*) index.php?$1 [PT,QSA]
```

Note that these rulesets can be uses in a .htaccess file, as well as in a <Directory> block.
Using mod_rewrite to control access

This document supplements the mod_rewrite reference documentation. It describes how you can use mod_rewrite to control access to various resources, and other related techniques. This includes many examples of common uses of mod_rewrite, including detailed descriptions of how each works.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Virtual hosts
- Proxying
- Using RewriteMap
- Advanced techniques and tricks
- When not to use mod_rewrite
Description:
The following technique forbids the practice of other sites including your images inline in their pages. This practice is often referred to as "hotlinking", and results in your bandwidth being used to serve content for someone else's site.

Solution:
This technique relies on the value of the HTTP_REFERER variable, which is optional. As such, it's possible for some people to circumvent this limitation. However, most users will experience the failed request, which should, over time, result in the image being removed from that other site.

There are several ways that you can handle this situation.

In this first example, we simply deny the request, if it didn't initiate from a page on our site. For the purpose of this example, we assume that our site is www.example.com.

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !www.example.com [NC]
RewriteRule \.(gif|jpg|png)$ - [F,NC]
```

In this second example, instead of failing the request, we display an alternate image instead.

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !www.example.com [NC]
RewriteRule \.(gif|jpg|png)$ /images/go-away.png [R,NC]
```

In the third example, we redirect the request to an image on some other site.

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !www.example.com [NC]
```
Of these techniques, the last two tend to be the most effective in getting people to stop hotlinking your images, because they will simply not see the image that they expected to see.

**Discussion:**

If all you wish to do is deny access to the resource, rather than redirecting that request elsewhere, this can be accomplished without the use of mod_rewrite:

```plaintext
SetEnvIf Referer example\.com localreferer
<FilesMatch \.jpg|png|gif$>
  Order deny,allow
  Deny from all
  Allow from env=localreferer
</FilesMatch>
```
Description:
In this recipe, we discuss how to block persistent requests from a particular robot, or user agent.

The standard for robot exclusion defines a file, /robots.txt that specifies those portions of your website where you which to exclude robots. However, some robots do not honor these files.

Note that there are methods of accomplishing this which do not use mod_rewrite. Note also that any technique that relies on the clients USER_AGENT string can be circumvented very easily, since that string can be changed.

Solution:
We use a ruleset that specifies the directory to be protected, and the client USER_AGENT that identifies the malicious or persistent robot.

In this example, we are blocking a robot called NameOfBadRobot from a location /secret/files. You may also specify an IP address range, if you are trying to block that user agent only from the particular source.

```
RewriteCond %{HTTP_USER_AGENT} ^NameOfBadRobot
RewriteCond %{REMOTE_ADDR} =123\.45\.67\.[8-9]
RewriteRule ^/secret/files/ - [F]
```

Discussion:
Rather than using mod_rewrite for this, you can accomplish the same end using alternate means, as illustrated here:

```
SetEnvIfNoCase User-Agent ^NameOfBadRobot goaway
<Location /secret/files>
Order allow,deny
```
As noted above, this technique is trivial to circumvent, by simply modifying the USER_AGENT request header. If you are experiencing a sustained attack, you should consider blocking it at a higher level, such as at your firewall.
Description:

We wish to maintain a blacklist of hosts, rather like hosts.deny, and have those hosts blocked from accessing our server.

Solution:

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

```bash
##
## hosts.deny
##
## ATTENTION! This is a map, not a list, even when we treat it as such.
## mod_rewrite parses it for key/value pairs, so at least a
## dummy value "-" must be present for each entry.
##
193.102.180.41 -
bsdti1.sdm.de -
192.76.162.40 -
```

Discussion:

The second RewriteCond assumes that you have HostNameLookups turned on, so that client IP addresses will be resolved. If that's not the case, you should drop the second RewriteCond, and drop the [ OR ] flag from the first RewriteCond.
Description:
Redirect requests based on the Referer from which the request came, with different targets per Referer.

Solution:
The following ruleset uses a map file to associate each Referer with a redirection target.

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

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

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

The map file lists redirection targets for each referer, or, if we just wish to redirect back to where they came from, a "-" is placed in the map:

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

http://badguys.example.com/bad/index.html -
http://badguys.example.com/bad/index2.html -
http://badguys.example.com/bad/index3.html
http://somewhere.example.com/
```
Dynamic mass virtual hosts with mod_rewrite

This document supplements the mod_rewrite reference documentation. It describes how you can use mod_rewrite to create dynamically configured virtual hosts.

mod_rewrite is not the best way to configure virtual hosts. You should first consider the alternatives before resorting to mod_rewrite. See also the "how to avoid mod_rewrite document.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Proxying
- RewriteMap
- Advanced techniques and tricks
- When not to use mod_rewrite
Virtual Hosts For Arbitrary Hostnames

Description:
We want to automatically create a virtual host for every hostname which resolves in our domain, without having to create new VirtualHost sections.

In this recipe, we assume that we'll be using the hostname www.SITE.example.com for each user, and serve their content out of /home/SITE/www.

Solution:

```
RewriteEngine on
RewriteMap lowercase int:tolower
RewriteCond %{lowercase:%{HTTP_HOST}} ^www\.[^.]\+.example\..com$
RewriteRule ^(.*) /home/%1/www$1
```

Discussion

You will need to take care of the DNS resolution - Apache does not handle name resolution. You'll need either to create CNAME records for each hostname, or a DNS wildcard record. Creating DNS records is beyond the scope of this document.

The internal tolower RewriteMap directive is used to ensure that the hostnames being used are all lowercase, so that there is no ambiguity in the directory structure which must be created.

Parentheses used in a RewriteCond are captured into the backreferences %1, %2, etc, while parentheses used in RewriteRule are captured into the backreferences $1, $2,
etc.

As with many techniques discussed in this document, mod_rewrite really isn’t the best way to accomplish this task. You should, instead, consider using `mod_vhost_alias` instead, as it will much more grace fully handle anything beyond serving static files, such as any dynamic content, and Alias resolution.
This extract from httpd.conf does the same thing as the first example. The first half is very similar to the corresponding part above, except for some changes, required for backward compatibility and to make the mod_rewrite part work properly; the second half configures mod_rewrite to do the actual work.

Because mod_rewrite runs before other URI translation modules (e.g., mod_alias), mod_rewrite must be told to explicitly ignore any URLs that would have been handled by those modules. And, because these rules would otherwise bypass any ScriptAlias directives, we must have mod_rewrite explicitly enact those mappings.

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

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

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

RewriteEngine On

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

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

## and now deal with CGIs - we have to force a handler
```
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteRule ^/(.*)$ /www/hosts/${lowercase:%{SERVER_NAME}}/cgi-bin/$1 [H=cgi-script]
This arrangement uses more advanced `mod_rewrite` features to work out the translation from virtual host to document root, from a separate configuration file. This provides more flexibility, but requires more complicated configuration.

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

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

The `httpd.conf` should contain the following:

```
RewriteEngine on
RewriteMap lowercase int:tolower

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

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

RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond %{lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond %{vhost:%1} ^/(.*)$
RewriteRule ^/(.*)$ %1/cgi-bin/$1 [H=cgi-script]
```
Apache HTTP Server Version 2.2

Apache > HTTP Server > Documentation > Version 2.2 > Rewrite
Using mod_rewrite for Proxying

This document supplements the mod_rewrite reference documentation. It describes how to use the RewriteRule's [P] flag to proxy content to another server. A number of recipes are provided that describe common scenarios.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Virtual hosts
- Using RewriteMap
- Advanced techniques and tricks
- When not to use mod_rewrite
Description:
mod_rewrite provides the [P] flag, which allows URLs to be passed, via mod_proxy, to another server. Two examples are given here. In one example, a URL is passed directly to another server, and served as though it were a local URL. In the other example, we proxy missing content to a back-end server.

Solution:
To simply map a URL to another server, we use the [P] flag, as follows:

```
RewriteEngine on
RewriteBase /products/
RewriteRule ^widget/(.*)$ http://product.example.com/widget/$1 [P]
ProxyPassReverse /products/widget/ http://product.example.com/widget/
```

In the second example, we proxy the request only if we can't find the resource locally. This can be very useful when you're migrating from one server to another, and you're not sure if all the content has been migrated yet.

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^/(.*) http://old.example.com/$1 [P]
ProxyPassReverse / http://old.example.com/
```

Discussion:
In each case, we add a ProxyPassReverse directive to ensure that any redirects issued by the backend are correctly passed on to the client.

Consider using either ProxyPass or ProxyPassMatch whenever possible in preference to mod_rewrite.
Using RewriteMap

This document supplements the mod_rewrite reference documentation. It describes the use of the RewriteMap directive, and provides examples of each of the various RewriteMap types.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Virtual hosts
- Proxying
- Advanced techniques and tricks
- When not to use mod_rewrite
The `RewriteMap` directive defines an external function which can be called in the context of `RewriteRule` or `RewriteCond` directives to perform rewriting that is too complicated, or too specialized to be performed just by regular expressions. The source of this lookup can be any of the types listed in the sections below, and enumerated in the `RewriteMap` reference documentation.

The syntax of the `RewriteMap` directive is as follows:

```plaintext
RewriteMap  MapName  MapType:MapSource
```

The `MapName` is an arbitrary name that you assign to the map, and which you will use in directives later on. Arguments are passed to the map via the following syntax:

```plaintext
\${ 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:

```plaintext
RewriteMap examplemap txt:/path/to/file/map.txt
```

You would then be able to use this map in a `RewriteRule` as follows:

```plaintext
RewriteRule  ^/ex/(.*)  \${examplemap:$1}
```
A default value can be specified in the event that nothing is found in the map:

```
RewriteRule ^/ex/(.*) ${examplemap:$1}/not_found.html
```

**Per-directory and .htaccess context**

The `RewriteMap` directive may not be used in `<Directory>` sections or `.htaccess` files. You must declare the map in server or virtualhost context. You may use the map, once created, in your `RewriteRule` and `RewriteCond` directives in those scopes. You just can't **declare** it in those scopes.

The sections that follow describe the various *MapTypes* that may be used, and give examples of each.
When a MapType of txt is used, the MapSource is a filesystem path to a plain-text mapping file, containing space-separated key/value pair per line. Optionally, a line may be contain a comment, starting with a '#' character.

For example, the following might be valid entries in a map file.

```
# Comment line
MatchingKey SubstValue
MatchingKey SubstValue # comment
```

When the RewriteMap is invoked the argument is looked for in the first argument of a line, and, if found, the substitution value is returned.

For example, we might use a mapfile to translate product names to product IDs for easier-to-remember URLs, using the following recipe:

**Product to ID configuration**

```
RewriteMap product2id txt:/etc/apache2/productmap.txt
RewriteRule ^/product/(.*) /prods.php?id=${product2id:$1|NOTFOUND} [PT]
```

We assume here that the prods.php script knows what to do when it received an argument of id=NOTFOUND when a product is not found in the lookup map.

The file /etc/apache2/productmap.txt then contains the following:

**Product to ID map**

```
##
## productmap.txt - Product to ID map file
##
```
Thus, when `http://example.com/product/television` is requested, the RewriteRule is applied, and the request is internally mapped to `/prods.php?id=993`.

**Note: .htaccess files**

The example given is crafted to be used in server or virtualhost scope. If you're planning to use this in a `.htaccess` file, you'll need to remove the leading slash from the rewrite pattern in order for it to match anything:

```plaintext
RewriteRule ^product/(.*) /prods.php?id=${product2id:$1|NOTFOUND} [PT]
```

**Cached lookups**

The looked-up keys are cached by httpd until the mtime (modified time) of the mapfile changes, or the httpd server is restarted. This ensures better performance on maps that are called by many requests.
When a MapType of `rnd` is used, the MapSource is a filesystem path to a plain-text mapping file, each line of which contains a key, and one or more values separated by `|`. One of these values will be chosen at random if the key is matched.

For example, you might use the following map file and directives to provide a random load balancing between several back-end server, via a reverse-proxy. Images are sent to one of the servers in the 'static' pool, while everything else is sent to one of the 'dynamic' pool.

### Rewrite map file

```
##
##  map.txt -- rewriting map
##

static www1|www2|www3|www4
dynamic www5|www6
```

### Configuration directives

```
RewriteMap servers rnd:/path/to/file/map.txt

RewriteRule ^/(.*\.(png|gif|jpg)) http://${servers:static}/$1 [NC,P,L]
RewriteRule ^/(.*) http://${servers:dynamic}/$1 [P,L]
```

So, when an image is requested and the first of these rules is matched, `RewriteMap` looks up the string `static` in the map file, which returns one of the specified hostnames at random, which is then used in the `RewriteRule` target.

If you wanted to have one of the servers more likely to be chosen (for example, if one of the server has more memory than the others, and so can handle more requests) simply list it more times in the map file.
static www1|www1|www2|www3|www4
When a MapType of dbm is used, the MapSource is a filesystem path to a DBM database file containing key/value pairs to be used in the mapping. This works exactly the same way as the txt map, but is much faster, because a DBM is indexed, whereas a text file is not. This allows more rapid access to the desired key.

You may optionally specify a particular dbm type:

```plaintext
RewriteMap examplemap dbm=sdbm:/etc/apache/mapfile.dbm
```

The type can be sdbm, gdbm, ndbm or db. However, it is recommended that you just use the `httxt2dbm` utility that is provided with Apache HTTP Server, as it will use the correct DBM library, matching the one that was used when httpd itself was built.

To create a dbm file, first create a text map file as described in the `txt` section. Then run `httxt2dbm`:

```plaintext
$ httxt2dbm -i mapfile.txt -o mapfile.map
```

You can then reference the resulting file in your `RewriteMap` directive:

```plaintext
RewriteMap mapname dbm:/etc/apache/mapfile.map
```

Note that with some dbm types, more than one file is generated, with a common base name. For example, you may have two files named `mapfile.map.dir` and `mapfile.map.pag`. This is normal, and you need only use the base name `mapfile.map` in your `RewriteMap` directive.

**Cached lookups**
The looked-up keys are cached by httpd until the \texttt{mtime}
(modified time) of the mapfile changes, or the httpd server is
restarted. This ensures better performance on maps that are
called by many requests.
When a MapType of `int` is used, the MapSource is one of the available internal RewriteMap functions. Module authors can provide additional internal functions by registering them with the `ap_register_rewrite_mapfunc` API. The functions that are provided by default are:

- **`toupper`**: Converts the key to all upper case.
- **`tolower`**: Converts the key to all lower case.
- **`escape`**: Translates special characters in the key to hex-encodings.
- **`unescape`**: Translates hex-encodings in the key back to special characters.

To use one of these functions, create a RewriteMap referencing the `int` function, and then use that in your RewriteRule:

```
Redirect a URI to an all-lowercase version of itself

RewriteMap lc int:tolower
RewriteRule (.*[A-Z]+.*) ${lc:$1} [R]
```

Please note that the example offered here is for illustration purposes only, and is not a recommendation. If you want to make URLs case-insensitive, consider using `mod_speling` instead.
When a MapType of prg is used, the MapSource is a filesystem path to an executable program which will providing the mapping behavior. This can be a compiled binary file, or a program in an interpreted language such as Perl or Python.

This program is started once, when the Apache HTTP Server is started, and then communicates with the rewriting engine via STDIN and STDOUT. That is, for each map function lookup, it expects one argument via STDIN, and should return one new-line terminated response string on STDOUT. If there is no corresponding lookup value, the map program should return the four-character string "NULL" to indicate this.

External rewriting programs are not started if they're defined in a context that does not have RewriteEngine set to on.

A simple example is shown here which will replace all dashes with underscores in a request URI.

**Rewrite configuration**

```
RewriteMap d2u prg:/www/bin/dash2under.pl
RewriteRule - ${d2u:%{REQUEST_URI}}
dash2under.pl
```

```
#!/usr/bin/perl
$| = 1; # Turn off I/O buffering
while (<STDIN>) {
    s/-/_/g; # Replace dashes with underscores
    print $_;
}
```

**Use a RewriteLock!**

When using a prg: RewriteMap, you should use a RewriteLock. Failure to do so will result in an error message
in the log file, and may result in a race condition on concurrent requests.

**Caution!**

- Keep your rewrite map program as simple as possible. If the program hangs, it will cause httpd to wait indefinitely for a response from the map, which will, in turn, cause httpd to stop responding to requests.
- Be sure to turn off buffering in your program. In Perl this is done by the second line in the example script: $| = 1;
  This will of course vary in other languages. Buffered I/O will cause httpd to wait for the output, and so it will hang.
- Remember that there is only one copy of the program, started at server startup. All requests will need to go through this one bottleneck. This can cause significant slowdowns if many requests must go through this process, or if the script itself is very slow.
The **RewriteMap** directive can occur more than once. For each mapping-function use one **RewriteMap** directive to declare its rewriting mapfile.

While you cannot **declare** a map in per-directory context (.htaccess files or <Directory> blocks) it is possible to **use** this map in per-directory context.
Advanced Techniques with mod_rewrite

This document supplements the mod_rewrite reference documentation. It provides a few advanced techniques and tricks using mod_rewrite.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Virtual hosts
- Proxying
- Using RewriteMap
- When not to use mod_rewrite
Description:
A common technique for distributing the burden of server load or storage space is called "sharding". When using this method, a front-end server will use the url to consistently "shard" users or objects to separate backend servers.

Solution:
A mapping is maintained, from users to target servers, in external map files. They look like:

```
user1 physical_host_of_user1
user2 physical_host_of_user2
...
```

We put this into a map.users-to-hosts file. The aim is to map:

```
/u/user1/anypath
```

to

```
http://physical_host_of_user1/u/user/anypath
```

thus every URL path need not be valid on every backend physical host. The following ruleset does this for us with the help of the map files assuming that server0 is a default server which will be used if a user has no entry in the map:

```
RewriteEngine on
RewriteMap users-to-hosts txt:/path/to/map.users-to-hosts
RewriteRule ^/u/([^/]+)/?(.*) http://$\{users-to-hosts:$1|server0\}/u/$1/$2
```
See the `RewriteMap` documentation for more discussion of the syntax of this directive.
Description:
We wish to dynamically generate content, but store it statically once it is generated. This rule will check for the existence of the static file, and if it's not there, generate it. The static files can be removed periodically, if desired (say, via cron) and will be regenerated on demand.

Solution:
This is done via the following ruleset:

```bash
# This example is valid in per-directory context
RewriteCond %{REQUEST_URI} !-U
RewriteRule ^(.+)\.html$ /regenerate_page.cgi [PT,L]
```

The -U operator determines whether the test string (in this case, REQUEST_URI) is a valid URL. It does this via a subrequest. In the event that this subrequest fails - that is, the requested resource doesn't exist - this rule invokes the CGI program `/regenerate_page.cgi`, which generates the requested resource and saves it into the document directory, so that the next time it is requested, a static copy can be served.

In this way, documents that are infrequently updated can be served in static form. If documents need to be refreshed, they can be deleted from the document directory, and they will then be regenerated the next time they are requested.
Load Balancing

Description:
We wish to randomly distribute load across several servers using mod_rewrite.

Solution:
We'll use RewriteMap and a list of servers to accomplish this.

```
RewriteEngine on
RewriteMap lb rnd:/path/to/serverlist.txt
RewriteRule ^/(.*) http://${lb:servers}/$1 [P,L]
```

`serverlist.txt` will contain a list of the servers:

```
## serverlist.txt
servers one.example.com|two.example.com|three.example.com
```

If you want one particular server to get more of the load than the others, add it more times to the list.

Discussion
Apache comes with a load-balancing module - `mod_proxy_balancer` - which is far more flexible and featureful than anything you can cobble together using mod_rewrite.
Description:
Wouldn't it be nice, while creating a complex web page, if the web browser would automatically refresh the page every time we save a new version from within our editor? Impossible?

Solution:
No! We just combine the MIME multipart feature, the web server NPH feature, and the URL manipulation power of mod_rewrite. First, we establish a new URL feature: Adding just :refresh to any URL causes the 'page' to be refreshed every time it is updated on the filesystem.

```
RewriteRule ^(/[uge]/[^/]+/?.*):refresh /internal/cgi/apache/nph-refresh?f=$1
```

Now when we reference the URL

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

this leads to the internal invocation of the URL

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

The only missing part is the NPH-CGI script. Although one would usually say "left as an exercise to the reader" ;-) I will provide this, too.

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

@pairs = split( /&/, $ENV{'QUERY_STRING'} )

foreach $pair (@pairs) {
    ( $name, $value ) = split( /=/, $pair );
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])//pack("C", hex($1))
    eval "\$\$name = "\$value";"
}

$QS_s = 1 if ( $QS_s eq '' )

$QS_n = 3600 if ( $QS_n eq '' )

if ( $QS_f eq '' ) {
    print "HTTP/1.0 200 OK\n"
    print "Content-type: text/html\n"
    print "&lt;b&gt;ERROR&lt;/b&gt;: No file given\n"
    exit(0);
}

if ( !-f $QS_f ) {
    print "HTTP/1.0 200 OK\n"
    print "Content-type: text/html\n"
    print "&lt;b&gt;ERROR&lt;/b&gt;: File $QS_f not found\n"
    exit(0);
}

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

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

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

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

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

sub mystat {
    local ($file) = $@[0];
    local ($time);
    ( $x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime ) = stat($f
return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ( $n = 0 ; $n & lt ; $QS_n ; $n++ ) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ( $mtime ne $mtimeL ) {
            $mtimeL = $mtime;
            sleep(2);
            $buffer = &readfile($QS_f);
            &print_http_headers_multipart_next;
            &displayhtml($buffer);
            sleep(5);
            $mtimeL = &mystat($QS_f);
            last;
        }
        sleep($QS_s);
    }
}

&print_http_headers_multipart_end;

exit(0);

###EOF###
Description:
Some sites with thousands of users use a structured homedir layout, *i.e.* each homedir is in a subdirectory which begins (for instance) with the first character of the username. So, 

/\sim larry/anypath is
/home/l/larry/public_html/anypath while
/\sim waldo/anypath is
/home/w/waldo/public_html/anypath.

Solution:
We use the following ruleset to expand the tilde URLs into the above layout.

```perl
RewriteEngine on
RewriteRule ^~/([a-z])[^a-z0-9]+(.*)
/home/$2/$1/public_html$3
```
Description:  
By default, redirecting to an HTML anchor doesn't work, because mod_rewrite escapes the # character, turning it into %23. This, in turn, breaks the redirection.

Solution:  
Use the [NE] flag on the RewriteRule. NE stands for No Escape.

Discussion:  
This technique will of course also work with other special characters that mod_rewrite, by default, URL-encodes.
Description:

We wish to use mod_rewrite to serve different content based on the time of day.

Solution:

There are a lot of variables named TIME_xxx for rewrite conditions. In conjunction with the special lexicographic comparison patterns <STRING, >STRING and =STRING we can do time-dependent redirects:

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

This provides the content of foo.day.html under the URL foo.html from 07:01-18:59 and at the remaining time the contents of foo.night.html.

**mod_cache**, intermediate proxies and browsers may each cache responses and cause the either page to be shown outside of the time-window configured. **mod_expires** may be used to control this effect. You are, of course, much better off simply serving the content dynamically, and customizing it based on the time of day.
Description:
At time, we want to maintain some kind of status when we perform a rewrite. For example, you want to make a note that you've done that rewrite, so that you can check later to see if a request can via that rewrite. One way to do this is by setting an environment variable.

Solution:
Use the [E] flag to set an environment variable.

```
RewriteEngine on
RewriteRule ^/horse/(.*) /pony/$1 [E=rewritten:1]
```

Later in your ruleset you might check for this environment variable using a RewriteCond:

```
RewriteCond %{ENV:rewritten} =1
```
When not to use mod_rewrite

This document supplements the mod_rewrite reference documentation. It describes perhaps one of the most important concepts about mod_rewrite - namely, when to avoid using it.

mod_rewrite should be considered a last resort, when other alternatives are found wanting. Using it when there are simpler alternatives leads to configurations which are confusing, fragile, and hard to maintain. Understanding what other alternatives are available is a very important step towards mod_rewrite mastery.

Note that many of these examples won't work unchanged in your particular server configuration, so it's important that you understand them, rather than merely cutting and pasting the examples into your configuration.

The most common situation in which mod_rewrite is the right tool is when the very best solution requires access to the server configuration files, and you don't have that access. Some configuration directives are only available in the server configuration file. So if you are in a hosting situation where you only have .htaccess files to work with, you may need to resort to mod_rewrite.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Virtual hosts
- Proxying
- Using RewriteMap
Advanced techniques and tricks
mod_alias provides the Redirect and RedirectMatch directives, which provide a means to redirect one URL to another. This kind of simple redirection of one URL, or a class of URLs, to somewhere else, should be accomplished using these directives rather than RewriteRule. RedirectMatch allows you to include a regular expression in your redirection criteria, providing many of the benefits of using RewriteRule.

A common use for RewriteRule is to redirect an entire class of URLs. For example, all URLs in the /one directory must be redirected to http://one.example.com/, or perhaps all http requests must be redirected to https.

These situations are better handled by the Redirect directive. Remember that Redirect preserves path information. That is to say, a redirect for a URL /one will also redirect all URLs under that, such as /one/two.html and /one/three/four.html.

To redirect URLs under /one to http://one.example.com, do the following:

```
Redirect /one/ http://one.example.com/
```

To redirect http URLs to https, do the following:

```
<VirtualHost *:80> ServerName www.example.com
  Redirect / https://www.example.com/
</VirtualHost>

<VirtualHost *:443> ServerName www.example.com
  # ... SSL configuration goes here
</VirtualHost>
```

The use of RewriteRule to perform this task may be appropriate if there are other RewriteRule directives in the same scope.
This is because, when there are Redirect and RewriteRule directives in the same scope, the RewriteRule directives will run first, regardless of the order of appearance in the configuration file.

In the case of the *http-to-https* redirection, the use of RewriteRule would be appropriate if you don't have access to the main server configuration file, and are obliged to perform this task in a `.htaccess` file instead.
The **Alias** directive provides mapping from a URI to a directory - usually a directory outside of your **DocumentRoot**. Although it is possible to perform this mapping with **mod_rewrite**, **Alias** is the preferred method, for reasons of simplicity and performance.

### Using Alias

```
Alias /cats /var/www/virtualhosts/felines/htdocs
```

The use of **mod_rewrite** to perform this mapping may be appropriate when you do not have access to the server configuration files. **Alias** may only be used in server or virtualhost context, and not in a `.htaccess` file.

Symbolic links would be another way to accomplish the same thing, if you have **Options FollowSymLinks** enabled on your server.
Although it is possible to handle virtual hosts with mod_rewrite, it is seldom the right way. Creating individual <VirtualHost> blocks is almost always the right way to go. In the event that you have an enormous number of virtual hosts, consider using mod_vhost_alias to create these hosts automatically.

Third-party modules such as mod_macro are also useful for creating a large number of virtual hosts dynamically.

Using mod_rewrite for virtualhost creation may be appropriate if you are using a hosting service that does not provide you access to the server configuration files, and you are therefore restricted to configuration using .htaccess files.

See the virtual hosts with mod_rewrite document for more details on how you might accomplish this if it still seems like the right approach.
RewriteRule provides the [P] flag to pass rewritten URIs through mod_proxy.

```
RewriteRule ^/?images(.*) http://imageserver.local/images$1 [P]
```

However, in many cases, when there is no actual pattern matching needed, as in the example shown above, the ProxyPass directive is a better choice. The example here could be rendered as:

```
ProxyPass /images/ http://imageserver.local/images/
```

Note that whether you use RewriteRule or ProxyPass, you'll still need to use the ProxyPassReverse directive to catch redirects issued from the back-end server:

```
ProxyPassReverse /images/ http://imageserver.local/images/
```

You may need to use RewriteRule instead when there are other RewriteRules in effect in the same scope, as a RewriteRule will usually take effect before a ProxyPass, and so may preempt what you're trying to accomplish.
Apache mod_rewrite Technical Details

This document discusses some of the technical details of mod_rewrite and URL matching.

See also

- Module documentation
- mod_rewrite introduction
- Redirection and remapping
- Controlling access
- Virtual hosts
- Proxying
- Using RewriteMap
- Advanced techniques and tricks
- When not to use mod_rewrite
The internal processing of this module is very complex but needs to be explained once even to the average user to avoid common mistakes and to let you exploit its full functionality.
First you have to understand that when Apache processes a HTTP request it does this in phases. A hook for each of these phases is provided by the Apache API. Mod_rewrite uses two of these hooks: the URL-to-filename translation hook which is used after the HTTP request has been read but before any authorization starts and the Fixup hook which is triggered after the authorization phases and after the per-directory config files (.htaccess) have been read, but before the content handler is activated.

So, after a request comes in and Apache has determined the corresponding server (or virtual server) the rewriting engine starts processing of all mod_rewrite directives from the per-server configuration in the URL-to-filename phase. A few steps later when the final data directories are found, the per-directory configuration directives of mod_rewrite are triggered in the Fixup phase. In both situations mod_rewrite rewrites URLs either to new URLs or to filenames, although there is no obvious distinction between them. This is a usage of the API which was not intended to be this way when the API was designed, but as of Apache 1.x this is the only way mod_rewrite can operate. To make this point more clear remember the following two points:

1. Although mod_rewrite rewrites URLs to URLs, URLs to filenames and even filenames to filenames, the API currently provides only a URL-to-filename hook. In Apache 2.0 the two missing hooks will be added to make the processing more clear. But this point has no drawbacks for the user, it is just a fact which should be remembered: Apache does more in the URL-to-filename hook than the API intends for it.

2. Unbelievably mod_rewrite provides URL manipulations in per-directory context, i.e., within .htaccess files, although these are reached a very long time after the URLs have been translated to filenames. It has to be this way because
.htaccess files live in the filesystem, so processing has already reached this stage. In other words: According to the API phases at this time it is too late for any URL manipulations. To overcome this chicken and egg problem mod_rewrite uses a trick: When you manipulate a URL/filename in per-directory context mod_rewrite first rewrites the filename back to its corresponding URL (which is usually impossible, but see the RewriteBase directive below for the trick to achieve this) and then initiates a new internal sub-request with the new URL. This restarts processing of the API phases.

Again mod_rewrite tries hard to make this complicated step totally transparent to the user, but you should remember here: While URL manipulations in per-server context are really fast and efficient, per-directory rewrites are slow and inefficient due to this chicken and egg problem. But on the other hand this is the only way mod_rewrite can provide (locally restricted) URL manipulations to the average user.

Don't forget these two points!
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.
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*. 

**Figure 1:** The control flow through the rewriting ruleset
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.
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:

```c
#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 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 	exttt{ldd(1)} and 	exttt{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 => /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 	exttt{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.

<table>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
<th>Thread Safe?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASpell/PSpell</td>
<td>?</td>
<td>Yes</td>
<td>Be careful about sharing a connection across threads.</td>
</tr>
<tr>
<td>Berkeley DB</td>
<td>3.x, 4.x</td>
<td>Yes</td>
<td>Both low-level and high-level APIs are thread-safe. However, high-level API requires thread-safe access to errno.</td>
</tr>
<tr>
<td>bzip2</td>
<td>Yes</td>
<td></td>
<td>Both low-level and high-level APIs are thread-safe. However, high-level API requires thread-safe access to errno.</td>
</tr>
<tr>
<td>cdb</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-Client</td>
<td>Perhaps</td>
<td></td>
<td>c-client uses strtok() and gethostbyname() are not thread-safe on most C library implementations. c-client's static data is meant to be shared. If strtok() and gethostbyname() are thread-safe on your OS, c-client may be thread-safe.</td>
</tr>
<tr>
<td>libcrypt</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expat</td>
<td>Yes</td>
<td></td>
<td>Need a separate parser instance</td>
</tr>
<tr>
<td>FreeTDS</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FreeType</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GD 1.8.x</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GD 2.0.x</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gdbm</td>
<td>No</td>
<td></td>
<td>Errors returned via a static gdbm_error</td>
</tr>
<tr>
<td>ImageMagick</td>
<td>5.2.2</td>
<td>Yes</td>
<td>ImageMagick docs claim it is thread-safe since version 5.2.2 (see Change log).</td>
</tr>
<tr>
<td>Imlib2</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>libjpeg</td>
<td>v6b</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>libmysqlclient</td>
<td>Yes</td>
<td></td>
<td>Use mysqlclient_r library variant</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information, please read <a href="http://dev.mysql.com/doc/mysql/en/Threaded_clients.html">http://dev.mysql.com/doc/mysql/en/Threaded_clients.html</a></td>
</tr>
<tr>
<td>Module</td>
<td>Version</td>
<td>Required</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ming</td>
<td>0.2a</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Net-SNMP</td>
<td>5.0.x</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>OpenLDAP</td>
<td>2.1.x</td>
<td>Yes</td>
<td>Use <code>ldap_r</code> library variant to ensure</td>
</tr>
<tr>
<td>OpenSSL</td>
<td>0.9.6g</td>
<td>Yes</td>
<td>Requires proper usage of <code>CRYPTO_set_id_callback</code></td>
</tr>
<tr>
<td>liboci8 (Oracle 8+)</td>
<td>8.x,9.x</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>pdflib</td>
<td>5.0.x</td>
<td>Yes</td>
<td>PDFLib docs claim it is thread-safe; it has been partially thread-safe since V1.91: <a href="http://www.pdflib.com/products/pdflib/">http://www.pdflib.com/products/pdflib/</a></td>
</tr>
<tr>
<td>libpng</td>
<td>1.0.x</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>libpng</td>
<td>1.2.x</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>libpq (PostgreSQL)</td>
<td>8.x</td>
<td>Yes</td>
<td>Don't share connections across threads and watch out for <code>crypt()</code> calls</td>
</tr>
<tr>
<td>Sablotron</td>
<td>0.95</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>zlib</td>
<td>1.1.4</td>
<td>Yes</td>
<td>Relies upon thread-safe <code>zalloc</code> and <code>zfree</code>; one way is to use <code>libc</code>'s <code>calloc/free</code> which are thread-safe.</td>
</tr>
</tbody>
</table>
Access Control

Access control refers to any means of controlling access to any resource. This is separate from authentication and authorization.
Access control can be done by several different modules. The most important of these is `mod authz_host`. Other modules discussed in this document include `mod setenvif` and `mod rewrite`. 
If you wish to restrict access to portions of your site based on the host address of your visitors, this is most easily done using `mod_authz_host`.

The `Allow` and `Deny` directives let you allow and deny access based on the host name, or host address, of the machine requesting a document. The `Order` directive goes hand-in-hand with these two, and tells Apache in which order to apply the filters.

The usage of these directives is:

**Allow from address**

where `address` is an IP address (or a partial IP address) or a fully qualified domain name (or a partial domain name); you may provide multiple addresses or domain names, if desired.

For example, if you have someone spamming your message board, and you want to keep them out, you could do the following:

**Deny from 10.252.46.165**

Visitors coming from that address will not be able to see the content covered by this directive. If, instead, you have a machine name, rather than an IP address, you can use that.

**Deny from host.example.com**

And, if you'd like to block access from an entire domain, you can specify just part of an address or domain name:

**Deny from 192.168.205**

*Deny from phishers.example.com moreidiots.example*  
*Deny from ke*
Using **Order** will let you be sure that you are actually restricting things to the group that you want to let in, by combining a **Deny** and an **Allow** directive:

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

Listing just the **Allow** directive would not do what you want, because it will let folks from that host in, in addition to letting everyone in. What you want is to let *only* those folks in.
mod_authz_host, in conjunction with mod_setenvif, can be used to restrict access to your website based on the value of arbitrary environment variables. This is done with the `Allow from env=` and `Deny from env=` syntax.

```
SetEnvIf User-Agent BadBot GoAway=1
Order allow,deny
Allow from all
Deny from env=GoAway
```

**Warning:**
Access control by User-Agent is an unreliable technique, since the User-Agent header can be set to anything at all, at the whim of the end user.

In the above example, the environment variable GoAway is set to 1 if the User-Agent matches the string BadBot. Then we deny access for any request when this variable is set. This blocks that particular user agent from the site.

An environment variable test can be negated using the `=!` syntax:

```
Allow from env=!GoAway
```
The [F] **RewriteRule** flag causes a 403 Forbidden response to be sent. Using this, you can deny access to a resource based on arbitrary criteria.

For example, if you wish to block access to a resource between 8pm and 6am, you can do this using **mod_rewrite**.

```bash
RewriteEngine On
RewriteCond %{TIME_HOUR} >20 [OR]
RewriteCond %{TIME_HOUR} <07
RewriteRule ^/fridge - [F]
```

This will return a 403 Forbidden response for any request after 8pm or before 7am. This technique can be used for any criteria that you wish to check. You can also redirect, or otherwise rewrite these requests, if that approach is preferred.
You should also read the documentation for `mod_auth_basic` and `mod_authz_host` which contain some more information about how this all works. `mod_authn_alias` can also help in simplifying certain authentication configurations.

See the [Authentication and Authorization](https://httpd.apache.org/docs/2.4/howto/) howto.